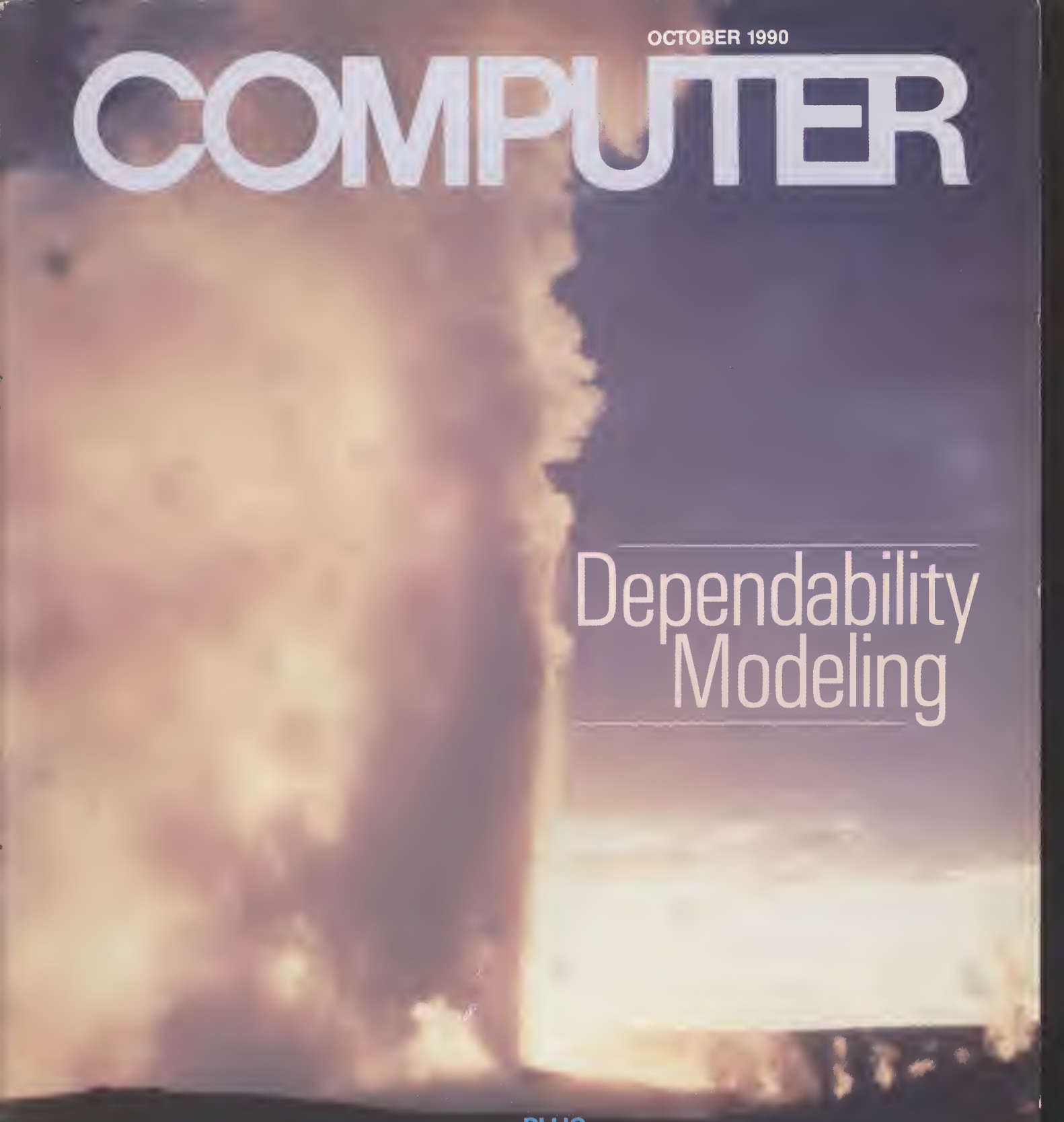
OCTOBER 1990

COMPUTER



Dependability Modeling

PLUS:

- Fault-Tolerant Concurrent Software
- Fault-Tolerant Clock Synchronization
- Built-In Self-Test RAM
- Network Chain Reactions
- Taulbee Survey

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

TELESOFT SDT. AN EARLY WARNING SYSTEM FOR REAL-TIME SYSTEMS ENGINEERS.

Creating large, complex, real-time systems has always been a risky proposition. TeleSoft SDT is a design tool that lowers risk and cost by allowing systems engineers to define, analyze, and simulate alternative system solutions. This provides an early warning of errors *before* time and energy is spent actually writing code. ☐ SDT supports the international standard Specification and Description Language (SDL), a formal description technique that specifies and describes the functional behavior of real-time, interactive and distributed software systems. The

integrated SDT tool set includes an OSF/Motif-based Graphical Editor, Analyzer and Converter, Simulator Generator, and Automatic Documentation Generator. ☐ For complete information on TeleSoft SDT, call us today. The earlier, the better.

AT&T's 5ESS System Switch was designed using SDL. Their experience indicates that the average programmer was 10 times more effective using SDL than the same programmer using assembly language. Used in conjunction with TeleSoft SDT, SDL not only saves time and money at the start of a project, but enables systems engineers to produce a higher quality product with lower operating and maintenance costs.

SDL IS REQUIRED BY ALL EUROPEAN NATIONAL TELECOM COMPANIES AS WELL AS SUCH INTERNATIONAL PROJ-

ECTS AS THE EUROPEAN FIBER OPTIC NETWORK FOR VOICE, DATA AND VIDEO (RACE), AND THE NEW EUROPEAN DIGITAL CELLULAR TELEPHONE NETWORK (GSM). SDL AND TELESOFT SDT ARE ALSO APPLICABLE TO COMPLEX, REAL-TIME SYSTEMS DEVELOPMENT PROJECTS IN AEROSPACE, DEFENSE AND MANUFACTURING OPERATIONS.

TELESOFT
5959 CORNERSTONE COURT WEST
SAN DIEGO, CA 92121-9891
PHONE: 1-619 457-2700
TELEFAX: 1-619 452-1334

TELESOFT

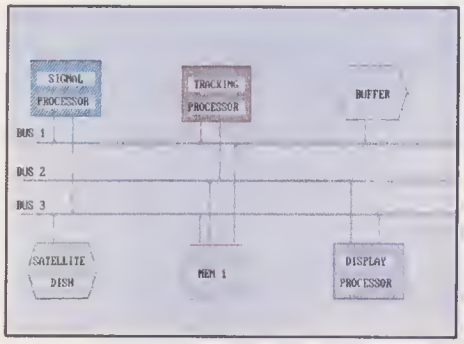
PROGRAMMED FOR PRODUCTIVITY

Reader Service Number 1

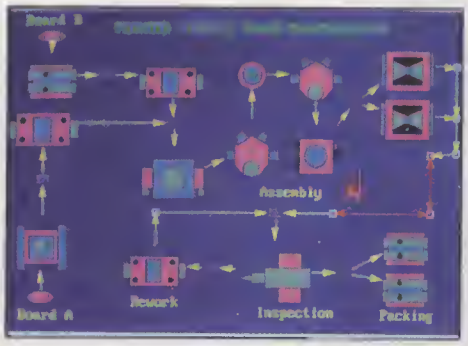
Breakthrough in presentation of simulation results
SIMSCRIPT II.5 with SIMGRAPHICS
Now you see an animated picture of the system

SIMGRAPHICS™ menu builder

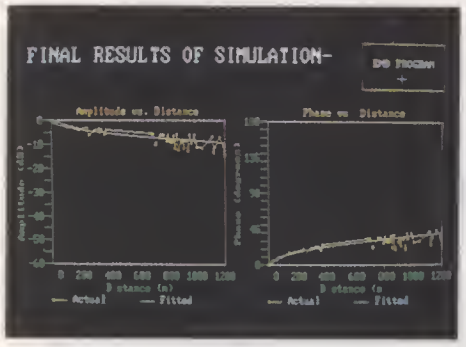
Draw your own icons--or use ours



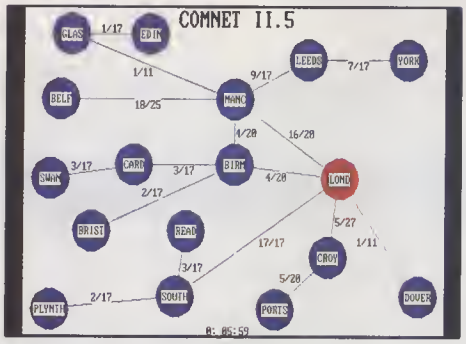
Local Area Network--NETWORK II.5®



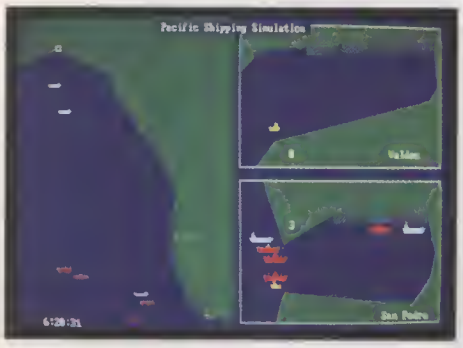
Factory in operation--SIMFACTORY II.5®



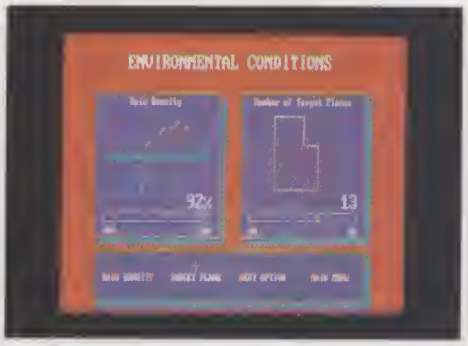
Presentation graphics



Communications--COMNET II.5™



Transportation system



SIMGRAPHICS advanced user interface

Free trial and training

See for yourself how simulation results are now easier to understand.

The free trial contains everything you need to try SIMGRAPHICS™ on your computer.

We send you SIMSCRIPT II.5, animated models, and complete documentation. You can build your own model or modify one of ours.

Try the SIMSCRIPT II.5® language, the timeliness of our support, the accuracy of our documentation, and the facilities for error checking--**everything you need for a successful project.**

For 28 years CACI has provided trial use of its simulation software--**no cost, no obligation.**

Act now for free training

For a limited time we will also include free training.

For immediate information

Call Doug Dittrich at (619) 457-9681. In Europe, call Nigel McNamara on (081) 332-0122. In Canada, call (613) 747-7467.

Rush information on SIMSCRIPT II.5 with SIMGRAPHICS

Limited offer-return the coupon today and we will also include one free course enrollment worth $950.

☐ Send details on your University Offer.

Name

Organization

Address

Telephone Fax

Computer Operating System

IEEE COMP

Return to:
CACI Products Company
3344 North Torrey Pines Court
La Jolla, California 92037, USA
Call Doug Dittrich at (619) 457-9681. Fax (619) 457-1184.

In Europe:
CACI Products Division
Palm Ct., 4 Heron Square
Richmond, Surrey TW9 1EW, United Kingdom
Call Nigel McNamara on (081) 332-0122. Fax (081) 332-0112.

In Canada:
CACI Products Company
1545 Carling Ave.
Ottawa, Ontario, K1Z 8P9
Call (613) 747-7467. Fax (613) 747-2224.

SIMSCRIPT II.5, NETWORK II.5, SIMFACTORY II.5, and SIMGRAPHICS are registered trademarks and service marks of CACI, INC. COMNET II.5 is a trademark and service mark of CACI, INC. ©1990 CACI, INC.

COMPUTER

October 1990 Published by the IEEE Computer Society Vol. 23, No. 10

ARTICLES

7 Dependability Modeling for Multiprocessors
Chita R. Das, Jeffrey T. Kreulen, Matthew J. Thazhuthaveetil, and Laxmi N. Bhuyan
In addition to delivering high performance, multiprocessors must be dependable in the presence of faults. This tutorial describes modeling techniques to assess multiprocessor dependability.

23 A System Architecture for Fault Tolerance in Concurrent Software
Massimo Ancona, Gabriella Dodero, Vittoria Gianuzzi, Andrea Clematis, and Eduardo B. Fernandez
This proposed architecture separates the application from the recovery software, giving programmers a single environment that lets them use the most appropriate fault-tolerance scheme.

33 Fault-Tolerant Clock Synchronization in Distributed Systems
Parameswaran Ramanathan, Kang G. Shin, and Ricky W. Butler
Software algorithms are suitable only where loose synchronization is acceptable, and hardware algorithms are expensive. A hybrid scheme achieves reasonably tight synchronization and is cost-effective.

45 Built-in Self-Testing of Random-Access Memories
Manoj Franklin and Kewal K. Saluja
An examination of BIST schemes indicates that approaches based on test architectures rather than on test algorithms are versatile and will likely predominate in the future.

57 Chain Reactions in Networks
Udi Manber
Theoretically, it's impossible to prevent the berserk loops of messages that can bring down an entire computer network. But a carefully applied set of rules will minimize these disasters.

SPECIAL REPORT

65 The 1988-89 Taulbee Survey Report
David Gries and Dorothy Marsh
The Computing Research Board's survey shows an 8 percent increase in PhDs granted. Of the 625 new PhDs, none are black, six are Hispanic, and 87 are women.

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380. IEEE Computer Society Headquarters, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription included in society member dues. Nonmember subscription rates: available upon request. Single copy prices: members $10.00; nonmembers $20.00. This magazine is also available in microfiche form.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08855. Second class postage is paid at New York, New York, and at additional mailing offices.

Copyright and reprint permission: Copyright © 1990 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US copyright law for private use of patrons: (1) those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress Street, Salem, MA 01970; (2) pre-1978 articles without fee. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *Computer*, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in *Computer* does not necessarily constitute endorsement by the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space considerations.

BPA

DEPARTMENTS

73 Standards
User interface management systems and application portability

76 Computer Society News
CS chapter formed in Moscow; 95 programs now CSAC accredited

79 Update
Microprocessor industry scrutinizes engineer's patent

82 Product Reviews
Postscript output on a Laserjet printer; a head-of-the-class spreadsheet

92 New Products

97 IC Announcements

98 Microsystem Announcements

100 Call for Papers

102 Calendar

116 Book Reviews

119 CS Magazines

Career Opportunities, 109; Computer Society Information, 120; Membership Application, 95; Change-of-Address Form, 75; Advertiser/Product Index, 96; Reader Service Card, 96A

Cover photo: Old Faithful geyser, Yellowstone National Park, courtesy US National Park Service

Cover design: Jay Simpson, Design & Direction

In the next issue

The Reactive Keyboard: A Predictive Typing Aid; Knowledge Bases and Databases: A Convergence of Technologies, A Divergence of Interests; Multiprocessor Algorithms for Relational Database Operators on Hypercube Systems; The Garnet User Interface Development Environment; A Functional Correctness Model of Program Verification

COMPUTER

10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264

Editor-in-Chief
Bruce D. Shriver, University of Southwestern Louisiana

Associate Technical Editor
Jon T. Butler, Naval Postgraduate School

Editorial Board
Dharma P. Agrawal, North Carolina State University
Dennis R. Allison, Stanford University
Fletcher Buckley, GE Aerospace
Scott Davidson, AT&T
Richard Eckhouse, MOCO
Scott E. Fahlman, Carnegie Mellon University
John R. Gurd, University of Manchester
Brent T. Hailpern, IBM T.J. Watson Research Center
James H. Haynes, University of California, Santa Cruz
Guy Johnson, Rochester Institute of Technology
Yale Patt, University of Michigan
Guylaine M. Pollock, Sandia National Laboratories
Martha Sloan, Michigan Technological University
Akinori Yonezawa, Tokyo Institute of Technology

Submissions
Send six double-spaced copies of articles and special-issue proposals to Jon T. Butler, Naval Postgraduate School, Electrical and Computer Engineering, Code EC/Bu, Monterey, CA 93943-5004, phone (408) 646-3299. For electronic submission, Butler's addresses are j.butler on Compmail II and butler@cs.nps.navy.mil on Internet.

Staff
Publisher: True Seaborn
Editorial Director: Marilyn Potes
Assistant Publisher: Douglas L. Combs
Assistant to the Publisher: Pat Paulsen

Managing Editor: Marilyn Potes
Staff Editors: Chuck Governale, Steve H. Wilcox, Bob Carlson
Contributing Editor: Ware Myers

Art Director: Jay Simpson
Design/Production: Toni Van Buskirk, Janice Garton, Chris Patterson
Membership/Circulation: Christina Champion
Advertising Coordinators: Heidi Rex, Marian Tibayan

Magazine Advisory Committee
Jon T. Butler (chair), B. Chandrasekaran, Manuel D'Abreu, James Farrell III, Joe Hootman, Anil Jain, Sushil Jajodia, Ted Lewis, True Seaborn, Bruce D. Shriver, Pradip K. Srimani, John Staudhammer

Publications Board
Sallie Sheppard (chair), Dharma P. Agrawal, Victor Basili, P. Bruce Berra, J. Richard Burke, Jon T. Butler, J.T. Cain, B. Chandrasekaran, David Choy, James Cross, Manuel D'Abreu, James J. Farrell III, Joe Hootman, Sushil Jajodia, Glen G. Langdon, Ted Lewis, Ming T. Liu, C.V. Ramamoorthy, True Seaborn, Bruce D. Shriver, John Staudhammer, Harold Stone, Steven L. Tanimoto, Joseph Urban, Ben Wah, Ron Williams

Requirements Analysis

Cadre's Unified CASE product family brings software design automation to the entire systems development lifecycle, from requirements analysis through to product test.
Requirements analysis extracts product function and behavior from fundamental system requirements. Workstation-based Teamwork and ADAS describe system requirements and evaluate tradeoffs between cost and performance.

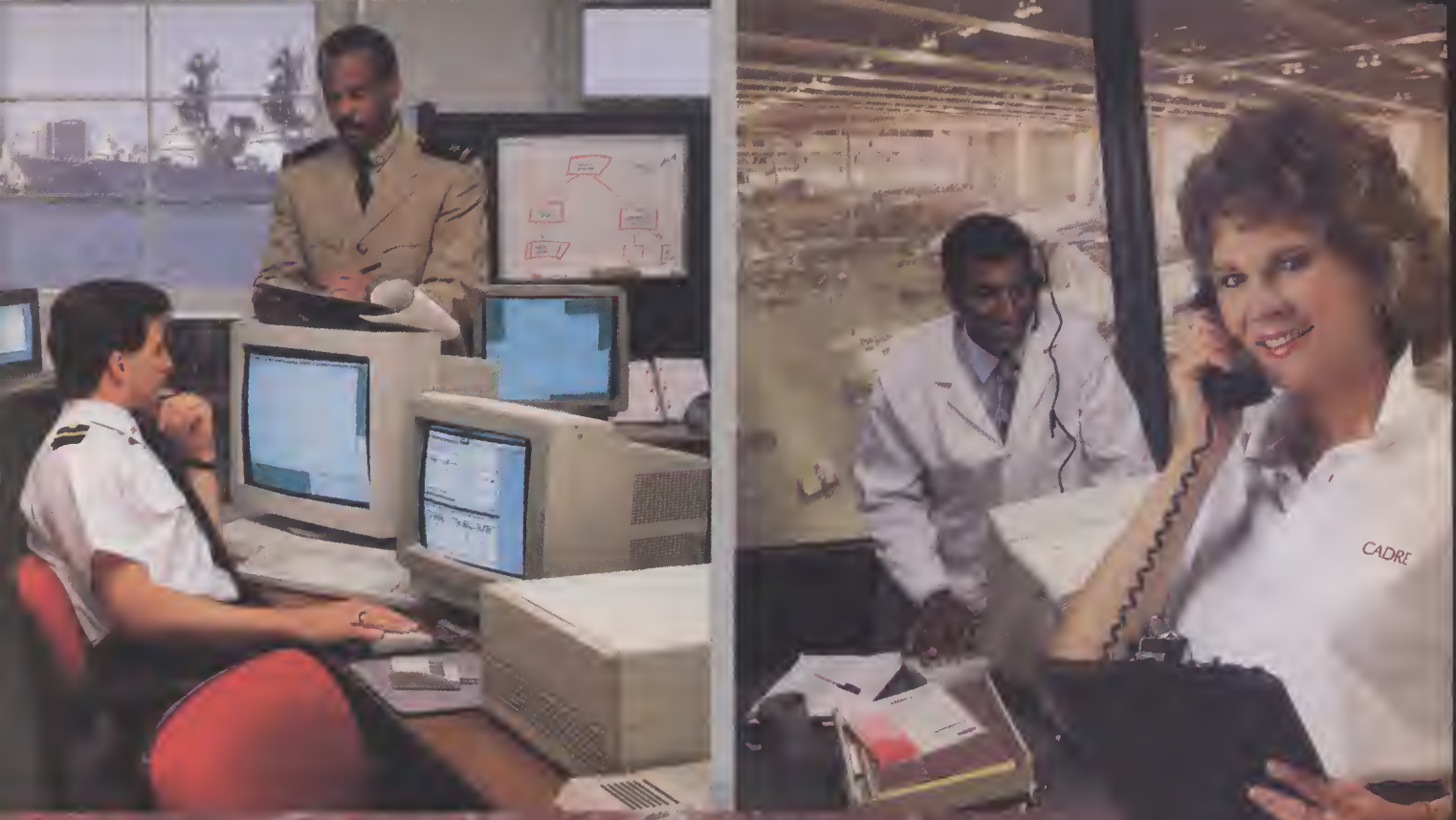
Design & Code

With requirements analysis completed, project teams implement software design and program logic using Teamwork. Design converts directly to code using Ada and C Source Builders, improving consistency and product maintainability. Cadre's networked Unified CASE environment moves design teams quickly through the product lifecycle to develop superior products.

teamwork®
Winning teams depend on it.

CADRE means

PathMap, PROBE, Software Analysis Workstation, Unified CASE, Ada Source Builder, and C Source Builder are trademarks of Cadre Tech



SE means cycle support

Test & Integration

Cadre's Unified CASE test products debug and test embedded software directly in target hardware. Test data automatically annotates original design for complete traceability.
Cadre's Software Analysis Workstation, or SAW, measures execution and performance of embedded software. PathMap, Cadre's Run-time Reverse Engineering tool, creates design-level Structure Charts from actual code executing in target hardware. PROBE in-circuit emulators allow software engineers to operate from their own workstation platforms.

Ongoing Product Support

The Unified CASE product family includes Cadre's responsive customer service, specialized training programs, consulting services, and ongoing product enhancements. How can Unified CASE improve your products? To find out

Call (401) 351-CASE

When critical systems need to be developed, project leaders call for Unified CASE solutions. Available only from Cadre.

Unified CASE™

CADRE

Cadre Technologies Inc.
222 Richmond Street, Providence, RI
02903 U.S.A. (401) 351-2273

Winning teams depend on us.

nologies Inc. Teamwork is a registered trademark of Cadre Technologies Inc. ADAS is a trademark of the Research Triangle Institute.

The joy of C-scape

Elegant graphics and text

The C-scape™ Interface Management System is a flexible library of C functions for data entry and validation, menus, text editing, context-sensitive help, and windowing. C-scape's powerful Look & Feel™ Screen Designer lets you create full-featured screens and automatically generates complete C source code.

C-scape includes easily modifiable high-level functions as well as primitives to construct new functions. Its object-oriented design helps you build more functional, more flexible, more portable, and more unique applications—and you'll have more fun doing it.

The industry standout. Many thousands of software developers worldwide have turned to the pleasure of C-scape. The press agrees: "C-scape is by far the best. . . . A joy to use," wrote IEEE *Computer*. Major companies have selected C-scape as a standard for software development.

C-scape's open architecture lets you use it with data base, graphics, or other C and C++ libraries. C-scape runs in text or graphics mode, so you can display text and graphics simultaneously. To port from DOS or OS/2 to UNIX, AIX, QNX, or VMS, just recompile. C-scape also supports Phar Lap and Rational DOS extenders.

Graphics. Run in color in text or graphics mode. Read images from PCX files.

Object-oriented architecture. Add custom features and create reusable code modules. C++ compatible.

Mouse support. Fully-integrated mouse support for menu selections, data entry fields, and to move and resize windows.

Portability. Hardware independent code. Supports DOS, OS/2, UNIX, AIX, VMS, others. Autodetects Hercules, CGA, EGA, VGA. Supports Phar Lap and Rational DOS extenders.

Text editing. Text editors with word wrap, block commands, and search and replace.

Field flexibility. Masked, protected, marked, required, no-echo, and named fields with complete data validation. Time, date, money, pop-up list, and many more higher-level functions; create your own.

Windows. Pop-up, tiled, bordered and exploding windows; size and numbers limited only by RAM.

Menus. Pop-up, pull-down, 123-style, or slug menus; create your own.

Context-sensitive help. Link help messages to individual screens or fields. Cross reference messages to create hypertext-like help.

Code generation. Build any type of screen or form with the Look & Feel™ Screen Designer, test it, then automatically convert it to C code.

Screen flexibility. Call screens from files at run time or link them in. Automatic vertical/horizontal scrolling.

International support. Offices in Berlin, Germany, with an international network of technical companies providing local training, support and consulting.

Trial with a smile. C-scape is powerful, flexible, portable, and easy to try. Test C-scape for 30 days. It offers a thorough manual and function reference, sample programs with source code, and an optional screen designer and source code generator. Oakland provides access to a 24-hour BBS, telephone services, and an international network of companies providing in-country support. No royalties, runtime licenses, runtime modules. After you register, you get complete library source code at no extra cost.

Call 800-233-3733 (617-491-7311 in Massachusetts, 206-746-8767 in Washington; see below for International). After the joy of C-scape, programming will never be the same.

DOS, OS/2 (Borland and Microsoft support): with Look & Feel, $499; library only, $399; UNIX, etc. start at $999; prices include library source. Training in Cambridge and Seattle each month. Mastercard and Visa accepted.

OAKLAND

Oakland Group, Inc. 675 Massachusetts Ave., Cambridge, MA 02139 USA. FAX: 617-868-4440. **Oakland Group, GmbH.** Alt Moabit 91-B, D-1000 Berlin 21, F.R.G. (030) 391 5045, FAX: (030) 393 4398. **Oakland International Technical Network** (training, support, consulting): **Australia** Noble Systems (02) 564-1200; **Benelux** TM Data (02159) 46814; **Denmark** Ravenholm (042) 887249; **Austria-Germany-Switzerland** ESM 07127/5244; **Norway** Ravenholm (02) 448855; **Sweden** Linsoft (013) 111588; **U.K.** Systemstar (0992) 500919. Photo by Jessica A. Boyatt; Kanji by Kaji Aso. Picture shows a C-scape program combining data entry with video images loaded from PCX files. C-scape and Look & Feel are trademarks of Oakland Group, Inc.; other trademarks belong to their respective companies. Copyright © 1990, by Oakland Group, Inc. Features, prices, and terms subject to change.

SURVEY & TUTORIAL SERIES

Dependability Modeling for Multiprocessors

Chita R. Das, Jeffrey T. Kreulen, and Matthew J. Thazhuthaveetil
Pennsylvania State University

Laxmi N. Bhuyan, Texas A&M University

In addition to delivering high performance, multiprocessors must be dependable in the presence of faults. This tutorial describes modeling techniques to assess multiprocessor dependability.

As multiprocessors take on more and increasingly critical tasks, quick and accurate evaluation of both the performance and the dependability of these parallel architectures becomes essential. However, while considerable research efforts have been directed at analyzing multiprocessor performance, dependability prediction has received less attention.

In general, increasing system dependability requires shifting resources from other design goals, such as high performance. Achieving both high computing power and high dependability makes system design very complex. However, through modeling, a system designer can assess whether a design meets dependability requirements, identify design bottlenecks, and select an optimal architecture.

018-9162/90/1000-0007$01.00 © 1990 IEEE

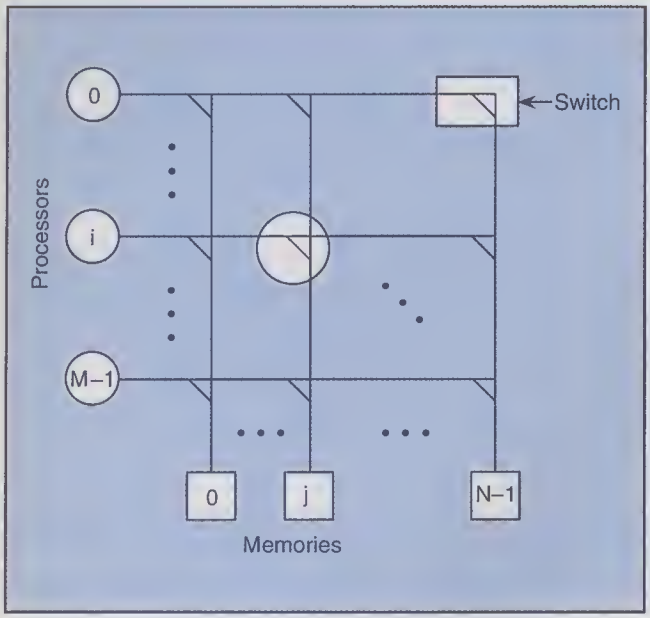


Figure 1. An $M\times N$ crossbar system.

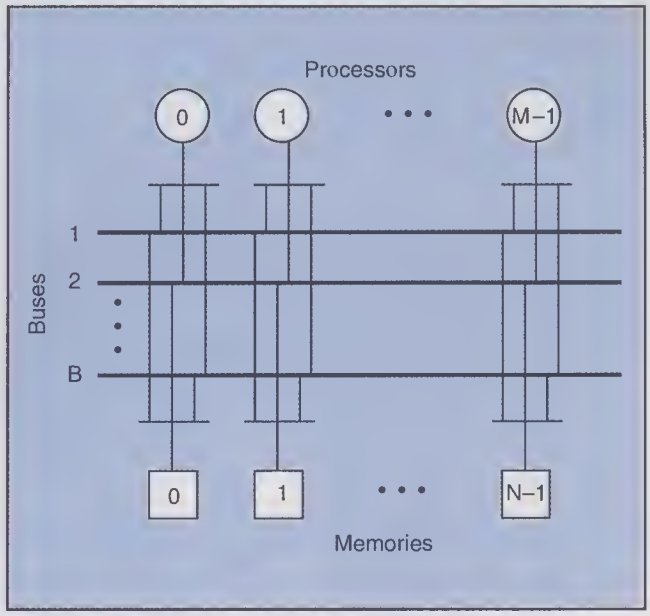


Figure 2. An $M\times N\times B$ multiple-bus multiprocessor.

Laprie and Coste have formally defined *dependability* as a high-level framework that encompasses measures like reliability and availability as distinct facets of system specification.[1] Based on a system's characteristics, either reliability or availability describes its dependability. For example, we can best characterize a nonrepairable system by its reliability, and a repairable system by its availability.

System *reliability* at time t is the probability that the system is operational during the interval $(0, t)$, given that it was operational at time 0. System *availability* at time t is the probability that the system is operational at the instant t. Unlike reliability, availability evaluation does not require continuous system operation during the interval $(0, t)$, since a system can go through the fail-repair cycle a number of times in this interval.

We need to define precisely the term "operational" as used in the above measures. A parallel machine can do useful work even if some of its components are nonfunctional, that is, it could be considered operational even given certain component failures. As we will discuss later, researchers have suggested four operational modes or models for dependability analysis. Each model defines a specific minimal requirement for a system to be considered operational.

Dependability measures do not reflect the performance of a multiprocessor system, only its operational status. However, designers must carefully consider the speed/dependability trade-off to ensure that higher speed is not attained at the cost of unacceptable dependability or vice versa. So, designers of parallel systems need performance-related dependability measures in addition to measures of dependability alone.

The quantitative evaluation of both dependability and performance-related dependability should be an integral part of multiprocessor design. While reliability and availability modeling are classic problems that have been applied to many areas of science and engineering, few efficient techniques have been devised for evaluating the dependability of parallel computers. This lack is mostly due to the complex nature of the interconnection networks that connect a multiprocessor's computing resources. Accurately modeling the interconnection network is extremely complex and can become an NP-complete problem. Different interconnection network topologies can require different modeling techniques, making the modeling of these systems a challenge.

This article is a tutorial on dependability and performance-related dependability models for multiprocessors. We will classify multiprocessors, explain some fundamental dependability modeling concepts, examine proposed reliability and availability models, discuss the status of research efforts on performance-related dependability, and illustrate the models' effectiveness with a few numerical examples. We will also provide a brief survey of software packages for dependability computation.

Multiprocessor classification

We can broadly divide multiprocessors into two categories: shared-memory architectures and distributed-memory architectures. In a shared-memory multiprocessor, the processors share a common, global memory, whereas each processor in a distributed-memory multiprocessor has a private memory.

A shared-memory multiprocessor consists of a number of processing elements connected to a number of globally addressable memory modules through an interconnection network. There are three main types of interconnection networks: crossbar, multiple-bus, and multistage interconnection networks.[2]

We call a multiprocessor with M processing elements and N memory modules an $M\times N$ system. Figure 1 depicts an $M\times N$ crossbar architecture with $M \cdot N$ crosspoint switches providing unique paths from each processing element to each memory module. The C.mmp multiprocessor is an example of a crossbar-based shared-memory

system. Since there are $M \cdot N$ switching elements in a crossbar, it is not an appropriate interconnection for building a large system.

Figure 2 shows an $M \times N \times B$ multiple-bus architecture with M processing elements, N memory modules, and B buses where $B \leq \min(M, N)$. Encore's Multimax and Sequent's Balance and Symmetry machines are examples of parallel processors using a single-bus ($B = 1$) interconnection. Bus-based shared-memory systems are more attractive to implement than crossbar-based systems, but they are suitable only for medium-sized systems with a few tens of processors.

A multistage interconnection network (MIN) is a cost-effective communication structure for large shared-memory organizations. An $N \times N$ MIN consists of $\log_a N$ stages of $a \times a$ switching elements, where a is a small integer (generally 2 or 4). Each stage of the network has N/a switches. A plethora of MIN topologies have been proposed that are functionally equivalent but that differ in their simultaneous interconnection capabilities. Figure 3 shows a specific type of MIN called a Butterfly network ($N = 16$ in the figure). Several commercial and experimental systems, such as the BBN Butterfly, the IBM RP3, and the University of Illinois' Cedar, have been developed or are being developed with this connection strategy.

The second type of multiprocessor, the distributed-memory architecture, consists of a network of computers (nodes), each with its own local memory. While interconnection topologies, such as rings, trees, and meshes, have been proposed for distributed-memory systems, the commercial hypercube machines (N-Cube/10, Intel iPSC, and the Connection Machine) have received much attention recently. Figure 4 shows a three-dimensional hypercube architecture.

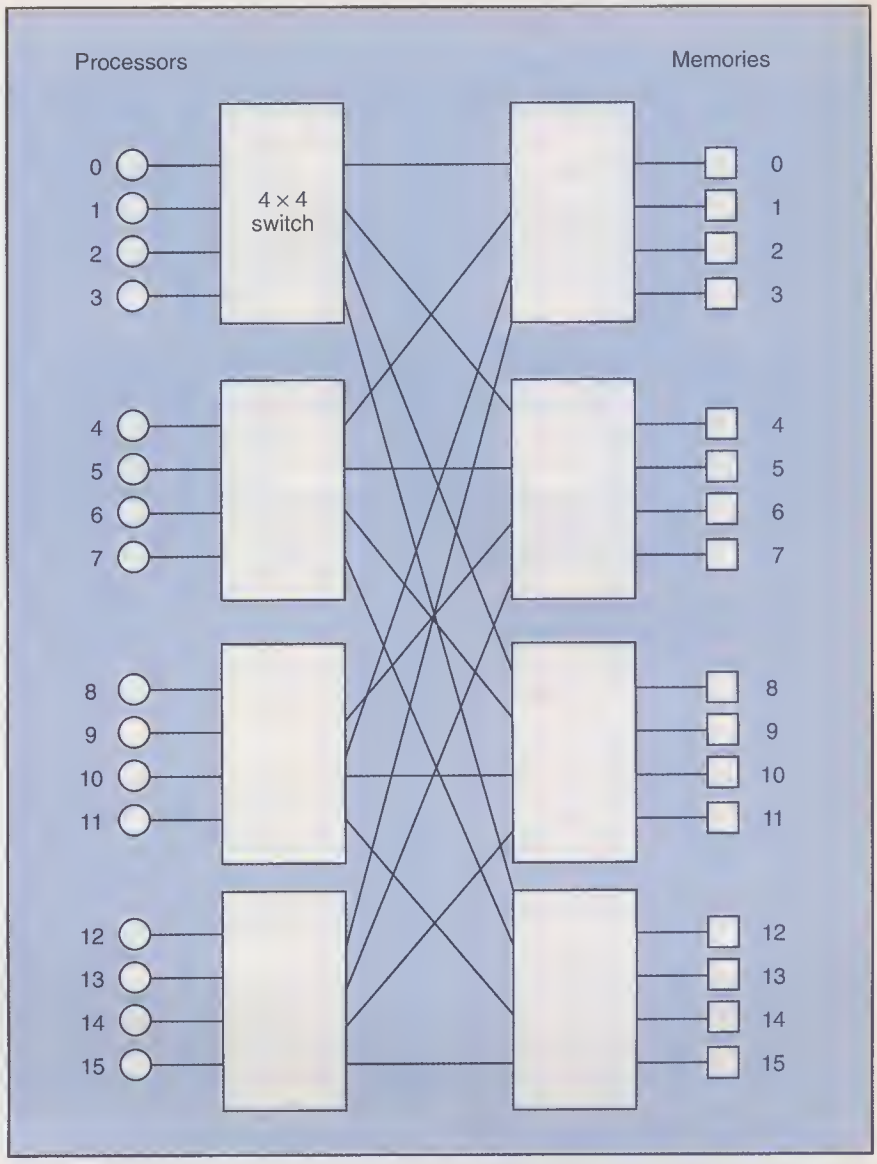


Figure 3. A 16×16 multistage interconnection network multiprocessor.

Preliminary steps

A model is an abstract representation of a system, derived using various assumptions about the system and involving three steps — definition, parameterization, and evaluation.

Definition. In this first step, the designer precisely defines the system and the objective of the modeling. The objective could be reliability, availability, or performance-related reliability or availability. The designer should also identify what constitutes an "operational" system, usually described in terms of available connectivity between working components (nodes) through the interconnection network. The minimum connectivity for an operational system could be based on

- a connection between a specific pair of input and output nodes (terminal dependability),
- connections between a known set of input and output nodes (multiterminal dependability),
- connections between a given percentage of the nodes in a system (task-based dependability), or
- connections between all the input and output nodes (network dependability).

Parameterization. The designer now defines the model's input and output pa-

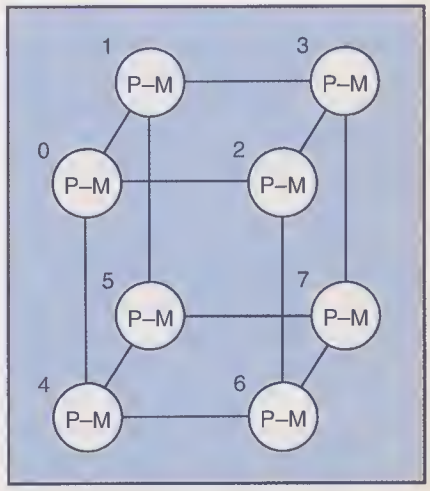


Figure 4. A three-dimensional hypercube. P-M stands for a processor-memory pair.

rameters. Typical input parameters include *component types*; *failure characteristics*, such as failure distribution, failure rate, and coverage (which is the probability that a system will reconfigure successfully following a fault); and *repair characteristics*, such as repair time distribution, repair rate, and probability of successful repair.

For mathematical simplicity, most dependability models assume that failure and repair distributions are exponential with constant rates. (The failure and repair rates of a component i are represented by λ_i and μ_i, respectively.) General distributions are used less frequently. However, simulation-based evaluation and combinatorial models can handle other appropriate distributions.

A model's output parameters include such measures as reliability, transient or steady-state availability, mean time to failure for nonrepairable systems, mean time between failures for repairable systems, reliability improvement factor, and performance-related dependability measures such as expected performance at time t or accumulated performance over a given time.

Evaluation. Finally, a designer can evaluate a model using either analytical techniques or simulation. Analytical techniques are usually quicker and more cost effective than simulation. However, analytical evaluation is generally less credible than simulation because of the simplifying assumptions used to develop a tractable analytical model. Simulation, on the other hand, can be conducted to an arbitrary level of detail.

We can divide analytical techniques into four categories: reliability block diagrams, fault trees, Markov models, and petri nets.

Reliability block diagrams or reliability graphs. This combinatorial technique represents in a graphical or block diagram form the various combinations of working components that exhibit successful overall system operation.

Fault trees. Fault tree representation is another combinatorial approach and is complementary to the reliability block diagram method. It represents system failure as a tree, where the root event is system failure, and it enumerates the different conditions that lead to failure. Reliability block diagrams and fault trees are equivalent in their representational power, and both are used for reliability analysis. Unfortunately, neither can model transient faults and on-line repair.

Markov models. Markov modeling is probably the most powerful technique for dependability analysis, since it can capture issues like on-line repair and transient faults in addition to the modeling capabilities of other techniques. A system's behavior is modeled as a finite-state continuous-time Markov chain, where each state represents a unique system configuration (which can be a working or failed state). A Markov chain can be cyclic or acyclic, depending on whether the system is repairable or not. A Markov model with n states is expressed with a set of n equations. Time-dependent (transient) system behavior can be obtained by solving n first-order differential equations, and steady-state behavior can be obtained by solving n linear equations. A summation of the working state probabilities yields reliability or availability. A number of software packages[3] (discussed later) use this technique to determine dependability.

Petri nets. This graphical approach represents the system as a directed graph. A system's stochastic nature is included in the model with timed transitions. The graph is then either solved using simulation or converted to a Markov model and solved numerically.

For multiprocessors, dependability modeling has centered around the two combinatorial techniques and Markov models. Petri nets have not been widely used due to the complexity of generating the graphical representation.

Reliability models

The four types of reliability evaluation techniques are terminal, multiterminal, task-based, and network reliability. Most dependability work has focused on crossbar, multiple-bus, MIN, and hypercube architectures, so we discuss each of these four multiprocessor types under each of the four reliability models.

Analytical models usually employ some simplifying assumptions to keep the analysis tractable. A common (and usually valid) assumption in dependability modeling is that components fail independently of each other, since each component is a separate entity. Another frequent assumption, exponentially distributed time to failure, facilitates system modeling.

Terminal reliability. The most basic reliability analysis is terminal reliability, the probability that at least one path exists between two specific terminals. Failure occurs when a terminal fails or when no path exists between the two terminals.

A terminal can be any of a range of entities. For example, each terminal could be a processor-memory pair, in which case terminal reliability could determine the reliability of two communicating processes. Alternatively, one terminal could be a processor and the other a memory module. Terminal reliability is more appropriate for communication networks than for parallel architectures, but it can be used for multiprocessors in which the source and destination nodes are uniquely defined.

To evaluate terminal reliability, we first assess the reliability of the individual components and enumerate their relative connectivity. Link failure between nodes is typically ignored here to simplify analysis. We can ascribe link failure to switch or node failure without losing generality. Also, since links are passive elements, their failure rate is almost negligible.

Crossbar systems. Consider the $M \times N$ crossbar system in Figure 1, where processing element PE_i requests connection to memory module MM_j. The hardware switch organization requires only a single crosspoint switch to connect an individual processor to an individual memory. Hence, to connect PE_i to MM_j, only the circled switch needs to be operational. Denoting switch reliability as $R_{sw}(t)$, terminal reliability is

$$TR(t) = R_p(t) \cdot R_m(t) \cdot R_{sw}(t)$$

where $R_p(t)$ and $R_m(t)$ are processor and memory reliability, respectively. For exponential distribution with failure rates λ_p and λ_m, $R_p(t) = e^{-\lambda_p t}$ and $R_m(t) = e^{-\lambda_m t}$. Similarly, $R_{sw}(t) = e^{-\lambda_{sw} t}$.

Bus-based systems. The bus-based system in Figure 2 has B alternate paths between any processor-memory pair, allowing communication between any processing element and any memory module as long as one bus is available. If a system has B buses, and the reliability of each bus is $R_b(t) = e^{-\lambda_b t}$, then

$$TR(t) = R_p(t) \cdot R_m(t) \cdot [1 - (1 - R_b(t))^B]$$

The last term is the probability that at least one bus is operational at time t.

MIN systems. There are two kinds of MINs: unique path and multiple path. For unique-path MINs, terminal reliability is related to the number of stages in the network. Given a unique-path MIN with n

(equal to $\log_a N$) stages of switching elements with individual switch reliability $R_{sw}(t)$,

$$TR(t) = R_p(t) \cdot R_m(t) \cdot R_{sw}^n(t)$$

For multiple-path MINs, there could be one or more paths connecting a processor to a memory, and the paths might not be disjoint. We can analyze such a system using a reliability block diagram to represent the system as a "series-parallel" structure. Consider two system components, for example, network switches A and B. If both switches are needed for a valid connection, then the switches are modeled as a series system with reliability expressed as $R_{series}(t) = R_A(t) \cdot R_B(t)$. If either switch can provide a valid connection, then they are in parallel, and $R_{parallel}(t) = [1 - (1 - R_A(t)) \cdot (1 - R_B(t))]$. This method quickly becomes intractable for larger systems.

A special issue of *IEEE Transactions on Reliability* contains a good collection of terminal reliability models.[4] For example, Varma and Raghavendra used terminal reliability to compare three classes of multiple-path networks. They obtained the terminal reliability expressions by iteratively analyzing reliability block diagrams, taking advantage of the networks' regular topologies in which all terminal connection pairs have the same terminal reliability.

Hypercube systems. Multiple-path complexity makes the analysis of hypercube systems similar to that of MIN systems. The best way to analyze an n-cube (2^n nodes) is to enumerate the paths and present them as a reliability block diagram. There are $n!$ nondisjoint alternate paths from each source node to each destination node.

Let's analyze the terminal reliability between nodes 0 and 7 for the eight-node hypercube in Figure 4. Figure 5 shows the reliability block diagram derived from this network. There are six possible nondisjoint paths from node 0 to node 7. If we assume all nodes have the same reliability ($R_n(t) = R_i(t)$ where $0 \le i \le 7$), the terminal reliability simplifies to

$$TR(t) = 6 \cdot R_n^4(t) - 6 \cdot R_n^5(t) - 3 \cdot R_n^6(t) + 6 \cdot R_n^7(t) - 2 \cdot R_n^8(t)$$

Multiterminal reliability. Multiterminal reliability is a logical extension of terminal reliability. It considers a specific subset of terminals instead of a specific pair. We define it as the probability that a given subset of terminals (nodes) are fully connected for simultaneous communication. This measure has much broader application than terminal reliability. For example, it can apply to the reliability of tasks on a nonreconfigurable system. Multiterminal reliability uses many of the same techniques and statistical assumptions as terminal reliability.

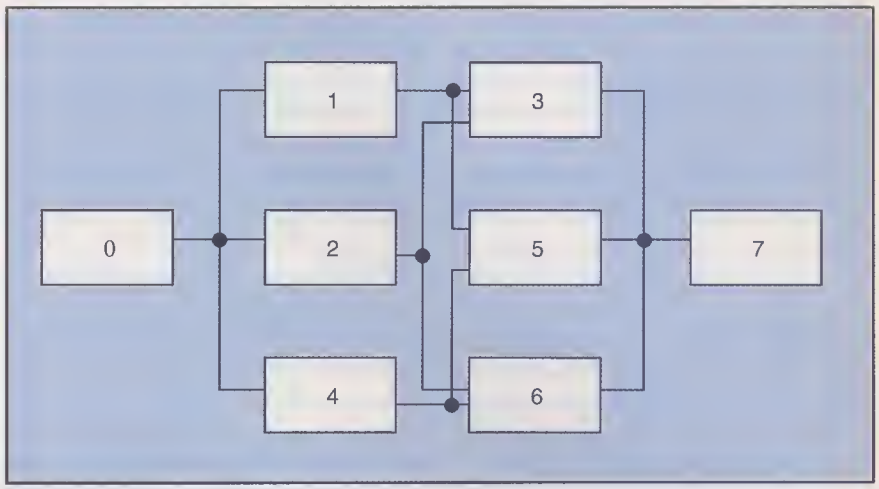


Figure 5. A reliability block diagram representation of various paths between node 0 and node 7.

Crossbar systems. An $M \times N$ crossbar system needs $x \cdot y$ operational crosspoint switches to connect x processors to y memories. This yields the system reliability

$$MTR(t) = R_p^x(t) \cdot R_m^y(t) \cdot R_{sw}^{x \cdot y}(t)$$

for $1 \le x \le M$, $1 \le y \le N$. The required components are uniquely specified, so the status of the other components does not affect system reliability.

Bus-based systems. For an $M \times N \times B$ bus-based system, the contribution of bus reliability is the same as in a terminal reliability scheme, since we need at least one bus. If a known set of x processors and y memories is required for a valid connection, reliability would be

$$MTR(t) = R_p^x(t) \cdot R_m^y(t) \cdot [1 - (1 - R_b(t))^B]$$

MIN systems. The multiterminal reliability of a unique-path MIN system is related to s, the number of switches required for the connection of x processing elements to y memory modules. This s depends on the location of the required processors and memory in the network. Once s is known, the multiterminal reliability becomes

$$MTR(t) = R_p^x(t) \cdot R_m^y(t) \cdot R_{sw}^s(t)$$

Multiterminal reliability analysis of multiple-path networks is complicated and requires more study.

Hypercube systems. Grnarov and Gerla[5] explored multiterminal reliability for distributed systems by applying a Boolean algebra technique. Their technique could apply to hypercube systems as well. A system is represented as a probabilistic graph $G(N, E)$, where N and E are the sets of nodes and links, respectively. Each component i is assigned a reliability. The multiterminal reliability algorithm uses an n-bit path identifier to represent each path between a source and destination, where n is the number of components that can fail. Hence, selecting $n = N$ considers only node failure, $n = E$ considers only link failure, and $n = N + E$ considers both node and link failure.

A bit in the path identifier is 1 if it is required for the connection, and x otherwise. All path identifiers for the input and output connections are then manipulated using Boolean algebra to find a final symbolic expression, which is converted to probabilistic form to determine multiterminal reliability.

Task-based reliability. Reliability models based on task requirements are more general than terminal or multiterminal reliability, since they do not explicitly specify the source and destination nodes. A system remains operational as long as a

task can be executed with the available resources. Therefore, if a task needs at least I processing elements and J memory modules in a shared-memory environment, or I nodes in a distributed-memory environment, the system is considered operational as long as those resources are available. Task-based reliability thus lets us improve fault tolerance by incorporating "graceful degradation" in the system.

Crossbar systems. Ingle and Siewiorek[6] used a combinatorial technique to find the C.mmp architecture's task-based reliability. Let's represent the probability that i out of x components are working at time t as

$$ {}_xC_i(t) = \binom{x}{i} R^i(t) \cdot (1 - R(t))^{x-i} \cdot C^{x-i} $$

where $R(t)$ is the component's reliability and C is the associated coverage factor. Using this notation, we express the $M \times N$ crossbar reliability that needs at least I processing elements and J memory modules as

$$ TBR(t) = \sum_{i=I}^{M} \sum_{j=J}^{N} {}_MC_i(t) \cdot {}_NC_j(t) \cdot R_{x-bar}(t) \quad (1) $$

where $M = N = 16$ for C.mmp. $R_{x\text{-}bar}(t)$ is the reliability of the complete 16×16 crossbar switch used in C.mmp. The two summations generate all possible processor-memory combinations.

This model does not consider crossbar degradation. We can formulate an approximate model for capturing crossbar degradation as follows. Assume we need a connected group of i processing elements and j memory modules in an $M \times N$ crossbar system. Since one switch is needed to connect each processing element to each memory module, we need $i \cdot j$ switching elements for the required connection. Let $\gamma = i \cdot j$, and let the unused switches $M \cdot N - \gamma = \varepsilon$. There are two ways an $M \times N$ crossbar system could provide the $i \times j$ required connection:

- when exactly i processing elements and j memory modules are functional, and the required γ switching elements are available to provide the needed connections (let's denote this probability as $P_{i,\,j(1)}(t)$, where (1) represents this first case); or
- when more than i processing elements or j memory modules are functional, but only γ switching elements are available to make the $i \times j$ connection — let's denote this probability as $P_{i,\,j(2)}(t)$.

We could write $P_{i,\,j(1)}(t)$ as

$$ P_{i,j\,(1)}(t) = {}_MC_i(t) \cdot {}_NC_j(t) \cdot R^{\gamma}_{sw}(t) \quad (2a) $$

Note that the condition of the extra switching elements ε is irrelevant when switch coverage is perfect. We can easily include imperfect coverage of switching elements in Equation 2a.

We can approximate the second term $P_{i,\,j(2)}(t)$ as

$$ P_{i,j\,(2)}(t) = \sum_{x}^{M} \sum_{y}^{N} {}_MC_x(t) \cdot {}_NC_y(t) \cdot R^{\gamma}_{sw}(t) \cdot (1 - R_{sw}(t))^{\varepsilon} \quad (2b) $$

where $x > i$ or $y > j$. In this equation, $(1 - R_{sw}(t))^{\varepsilon}$ denotes the probability that the extra switching elements have failed, so that there is only a connected group of $i \times j$.

We can then combine these two terms to define $TBR(t)$ as

$$ TBR(t) = \sum_{i=I}^{M} \sum_{j=J}^{N} (P_{i,j\,(1)}(t) + P_{i,j\,(2)}(t)) \quad (2c) $$

Bus-based systems. In a bus-based system (as in Figure 2), all computing resources are connected to the buses. Analysis is simple because there is a path between any processing element and memory module as long as there is at least one operational bus. The bus structure's contribution to system reliability is $1 - (1 - R_b(t))^B$. Substituting this term for $R_{x\text{-}bar}(t)$ in Equation 1 gives us the multiple-bus reliability.

MIN systems. Task-based reliability analysis for MIN multiprocessors is extremely difficult, mainly because of the complexity of the MIN. A detailed single-level Markov model of the complete system (including processing elements, memory modules, and switching elements) is almost impossible to construct due to the large number of states. Therefore, other techniques such as simulation and system decomposition have been used for task-based reliability analysis.

We have developed a simulation model for analyzing the task-based reliability of MIN multiprocessors. It considers the failure of processors, memory, and switching elements. The model can compute the task-based reliability of different types of unique-path and multiple-path MINs and could be used to validate analytical models. A drawback of this approach is that the simulation is expensive for large systems. However, we doubt that accurate analytical techniques can be developed for analyzing the task-based reliability of multiple-path systems.

Hierarchical decomposition can be used to analyze MIN systems. This scheme, which decomposes a larger system into smaller subsystems for ease of analysis, has been applied to the task-based reliability modeling of a Butterfly network-based multiprocessor. The Butterfly network uses 4×4 crossbar switches, as shown in Figure 3. The number of system nodes is 4^n, and each node consists of a processing element and a memory module. We call this a $4^n \times 4^n$ multiprocessor. We can partition a $4^n \times 4^n$ configuration into four $4^{n-1} \times 4^{n-1}$ subsystems without disturbing the connections. Therefore, the reliability of a 4^n multiprocessor depends on four 4^{n-1} subsystems and the connection pattern between them. The analysis starts with the probability expressions for a configuration of four processing elements, four memory modules, and two switching elements.

Let $P_{4(i,\,j)}(t)$ be the probability that i processing elements and j memory modules ($0 \le i, j \le 4$) are working at time t. We obtain this using simple combinatorics. The 4×4 analysis is used to find the probability that I processing elements and J memory modules are working in a 16×16 multiprocessor. We express this using four groups of 4×4 subsystems as

$$ P_{16(I,\,J)}(t) = P_{4(i_0,\,j_0)}(t) \cdot P_{4(i_1,\,j_1)}(t) \cdot P_{4(i_2,\,j_2)}(t) \cdot P_{4(i_3,\,j_3)}(t) $$

such that $(i_0 + i_1 + i_2 + i_3) = I$ and $(j_0 + j_1 + j_2 + j_3) = J$. All possible combinations of the I processing elements and J memory modules are generated for the above expression. Finally, the reliability of the 16-node system with at least I processing elements and J memory modules working is

$$ TBR(t) = \sum_{i=I}^{16} \sum_{j=J}^{16} P_{16(i,j)}(t) \quad (3) $$

We can obtain the reliability of a 64-node system from four 16-node configurations using the above technique. A 64-node Butterfly has three stages with 16 switching elements per stage. The first- and last-stage switching elements are included in the basic 4×4 analysis. Therefore, we can determine the required number of middle-stage switching elements for any working group by using an approximate calculation. The reliability of these switching elements is multiplied with the $P_{64(i,\,j)}(t)$ expression to guarantee that the middle-stage switching elements provide the required $i \times j$ connection for the two possible cases discussed in the earlier crossbar analysis. Similarly, we can compute the reliability of a 256-

node Butterfly multiprocessor using the results of four 64-node subsystems.

Hypercube systems. Kim et al. have proposed a task-based reliability model for hypercube architectures.[4] Their model, which finds the probability that an n-cube has at least I connected working nodes, is based on a decomposition technique and can handle any amount of system degradation. Since an n-cube can be divided into two $(n-1)$-cubes, this scheme recursively decomposes a large cube into smaller cubes until it is simple enough to model. The smallest cube is called the "base model" of the analysis. A 2-cube (four nodes) or a 3-cube (eight nodes) can be the base model, since in both cases it is simple to find exact expressions for I connected nodes. We can then recursively obtain the reliability of higher dimension cubes from either base model by approximating the connectivity between two lower dimension subcubes.

For example, to calculate the probability of j connected nodes in an n-cube, we divide the n-cube into two $(n-1)$-cubes. There are two cases in which an n-cube will have j connected nodes. In the first, exactly j connected nodes are working in the n-cube. In the second, more than j nodes (say x) are working, but the actual connectivity is only j. We can divide the j or x working nodes between the two $(n-1)$-cubes as k and $j-k$ or $x-k$. Five working node distributions are then analyzed from these two cases to find $P_j(t)$, the probability of j connected nodes from the two $(n-1)$-cubes. The n-cube reliability is

$$TBR(t) = \sum_{j=I}^{2^n} P_j(t) \qquad (4)$$

This approach could be extended to find a subcube of dimension m, for $m \leq n$, in an n-cube, based on the assumption that a job needs a cubic allocation instead of simply I connected nodes.

Network reliability. Network reliability is the probability of maintaining a full-access network. If we consider only unique-path networks, network reliability is the probability that all system processors and memories are connected (no degradation of the computing elements is allowed). A network with multiple communication paths (as in a multiple-bus architecture) can have full access even if network components fail.

Determining network reliability for crossbar, multiple-bus, and unique-path MIN architectures is trivial. All the $M \cdot N$ switches in a crossbar and all the $N/a \log_a N$ switches in a MIN must work to connect any input and any output. In a multiple-bus system, a path exists as long as at least one bus is working.

However, the analysis of multiple-path MINs is extremely difficult. Blake and Trivedi[4] have provided a bounding analysis for multiple-path MINs using a reliability block diagram approach. In this approach, the number of switching elements required for full connectivity is represented as one path (subsystem). The number of such paths is the same as the number of disjoint paths between the system's input and output nodes. We can than compute the MIN's reliability and mean time to failure using the standard reliability block diagram solution technique.

Network reliability is a special case of task-based reliability where the system does not allow the degradation of computing resources. We can use the task-based reliability models discussed above to determine network reliability.

Availability models

Reliability prediction gives us a pessimistic view of a system, since it does not consider repair activities. In gracefully degrading systems, for example, the repair of failed components increases the system's operational life and provides higher system availability.

Evaluating the availability of a complex system with a single (limited) repair facility is more complex than reliability modeling. This complexity is attributable to two factors. First, we must generate a system's Markov chain to analyze its availability. This process can be quite complex for a large system. The number of states in the model can also be too large to handle easily. Second, finding a closed-form solution for transient or steady-state availability is extremely difficult. Therefore, numerical techniques are used to compute availability.

There has been little research on analyzing multiprocessor availability. Research has focused mainly on task-based availability, typically based on the assumption that each type of component has a separate repair facility with an exponential distribution of repair time.

Crossbar systems. Smith and Trivedi[7] developed a Markov chain for a 16×16 crossbar system (C.mmp) where the entire crossbar switch is treated as a single element. This model is simple, since the crossbar failure results in system failure. The model is a two-dimensional Markov chain, one dimension representing processing element failure and the second representing memory module failure. The Markov chain could include separate repair of processing elements and memory modules and could then be solved with a software package.

Bus-based systems. Das and Bhuyan[8] have modeled a multiple-bus multiprocessor as a regular three-dimensional Markov chain where each dimension represents the degradation of one type of component (processor, memory, and bus). Assuming that the processors, memory modules, and buses each have independent repair facilities, the Markov chain can include the repair rates to depict a repairable system. The model can then be solved with a software package.

MIN systems. The availability model for MIN multiprocessors proposed by Das, Tien, and Bhuyan[9] can handle large systems and any level of degradation. This scheme uses a system decomposition approach to avoid the generation of a detailed single-level Markov chain. The model assumes independent failure and repair behavior of the processing elements, memory modules, and switching elements, and it conceptually divides the system into three subsystems connected in series. We can therefore develop independent Markov chains for the three subsystems.

Each subsystem's Markov diagram is a simple one-dimensional chain. For example, Figure 6 shows the Markov chain for a processor subsystem that needs at least I processing elements. λ_p and μ_p represent a processing element's failure rate and repair rate, respectively. The system moves from a state i to neighboring state $i-1$ with a failure rate $i \cdot \lambda_p$ or to state $i+1$ with repair rate μ_p. We have omitted the imperfect coverage and imprecise repair parameters for simplicity. The above model could be solved with a numerical analysis package such as the Hybrid Automated Reliability Predictor[3] to find the probability of i working processing elements $P_i(t)$, for $I \leq i \leq N$.

The Markov model for the memory subsystem is almost identical, except that the system tolerates only up to J failures. The memory failure and repair rates are λ_m and μ_m. Solving this model gives $P_j(t)$, the probability that there are j working memory modules at any time t.

The Markov chain for the switching elements is based on finding the number of

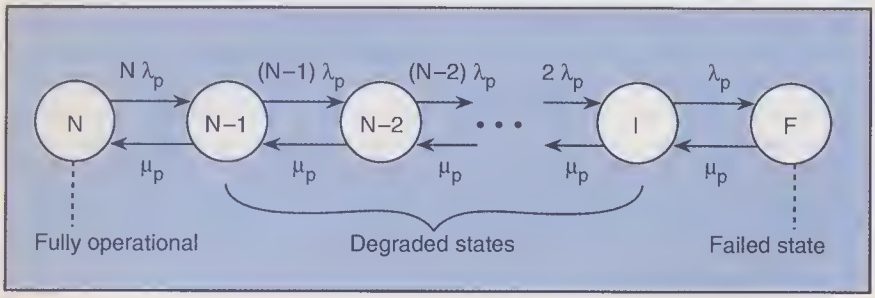


Figure 6. A Markov chain for a processor subsystem.

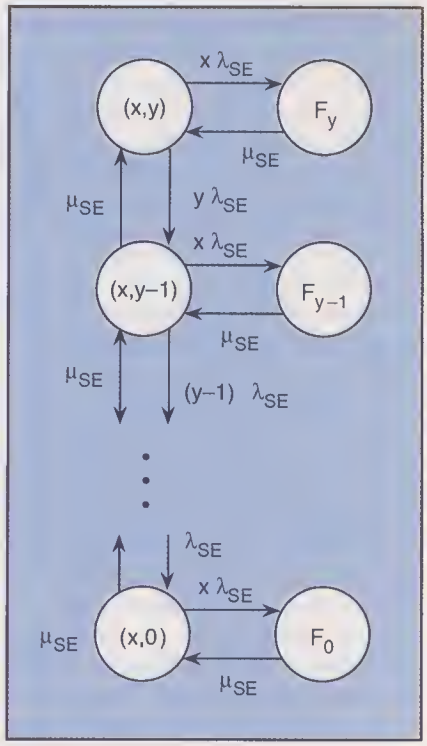


Figure 7. A Markov chain representation of a multistage interconnection network.

switching elements required for an $i \times j$ connection. Let's assume we need to compute the probability that eight processing elements and eight memory modules are connected in the 16×16 system shown in Figure 3. We could divide this system into four groups, each comprising four processing elements and four memory modules. We can distribute the eight processing elements and eight memory modules in two, three, or four groups.

There are two cases that could yield an 8×8 connection. In the first case, exactly eight processing elements and eight memory modules are working and are connected through the required number of switching elements. The other switching elements do not affect the connectivity. For example, if the working components are distributed in two groups, four switching elements must work for the connection to be valid. These switching elements belong to the two working groups and can be separated from the other four switching elements. The connectivity is 8×8 regardless of whether the other four switching elements work. The Markov chain in Figure 7 shows this case, with λ_{SE} and μ_{SE} representing the switching elements' failure and repair rates.

In Figure 7, (x, y) represents the required switching elements (x) and extra switching elements (y), such that $x + y$ is the total number of switching elements in the MIN. The horizontal transitions show that the system fails (state F_i, where $0 \le i \le y$) when any of the x required switching elements fails. The vertical transitions show the failure and repair of the extra y switching elements. Consequently, MIN reliability is $\Sigma_{j=0}^{y} P_{x,j}(t)$. Let's denote this as $P_{SE(1)}(t)$, where (1) indicates the first case, that is, exactly i processing elements and j memory modules are working. For Figure 7, with $x = y = 4$, $P_{SE(1)}(t) = (P_{4,4}(t) + P_{4,3}(t) + P_{4,2}(t) + P_{4,1}(t) + P_{4,0}(t))$. The exact $i \times j$ connection probability for the crossbar is $P_{i,j(1)}(t) = P_i(t) \cdot P_j(t) \cdot P_{SE(1)}(t)$.

In the second case, more than i processing elements (say α) or more than j memory modules (say β) can work while actual connectivity remains $i \times j$. For example, more than eight processing elements and eight memory modules can work in Figure 3, with connectivity remaining 8×8 as long as only the switching elements required for eight processing elements and eight memory modules are working. If the 8×8 connection is distributed in two groups, we need four switching elements. The other four switching elements should fail (making $y = 0$) so that the extra processing elements and memory modules in the rest of the system do not contribute to availability. Hence, $P_{SE(2)}(t) = P_{x,0}(t)$ is the probability that only x switching elements are working. We obtain the second-case probability $P_{i,j(2)}(t)$ by multiplying $P_{x,0}(t)$ by the probabilities that α processing elements are working and that β memory modules are working, given as $P_\alpha(t)$ and $P_\beta(t)$. We get the values for $P_\alpha(t)$ and $P_\beta(t)$ by solving the processor and memory Markov chains. We then determine system availability by adding all the working configuration probabilities using Equation 2c. We can apply this same approach to larger systems.

Hypercube systems. Das and Kim[10] have developed an availability model for hypercube computers with a single repair facility. It computes the transient and steady-state availability of hypercubes of any size. This approach does not require the generation of a hypercube's detailed Markov chain.

The model first computes the probability that any x nodes ($x \le N = 2^n$) in an n-cube are working (irrespective of their connection). This problem reduces to a machine repairman model (with or without imperfect coverage and imprecise repair) with N nodes. We can then easily compute the probability of a state x at any time t using any standard availability package.[3] Multiplying this probability by a connection probability yields the probability that j out of x working nodes are connected in a hypercube. The connection probability is computed using an approximation technique. A recursive equation is derived to compute this connection probability of an n-cube from a 2-cube base model.

We can explain the technique with an example of a 2-cube system. Figure 8a shows the Markov chain of a hypercube with no failure rate λ and no repair rate μ. We again assume that the coverage and repair processes are perfect. There are two possible states when two nodes are working. The first is state 2, in which both working nodes are connected. The second is state 1:1, where the two nodes are disconnected. This is possible when the diagonal nodes of a 2-cube are working. We can solve this Markov chain by using a package like the Hybrid Automated Reliability Predictor[3] to find the probability of any state j. An alternate approach is to compute this probability from the machine repairman model.[10]

If topological connectivity is neglected, the Markov chain of the system with N nodes is given by a machine repairman

model. As shown in Figure 8b, the model assumes the system is fully connected with N nodes and a single repairman. Let $P_{xc}(t)$ represent the probability of x working nodes at time t. The c indicates a fully connected system (it does not consider hypercube connectivity). If we analyze both Figure 8a and 8b, we can see that state 2 in Figure 8b represents a situation where two nodes are working in the system. In Figure 8a, the same two working nodes give two distinct states (2, 1:1) due to topological connection.

There is a fixed probability with which two working nodes in the 2-cube can be in state 2 or in state 1:1, respectively. For example, the probability that the two nodes will be connected (in state 2) is 2/3; the probability that they will be disconnected (in state 1:1) is 1/3. Let's call these probabilities "connection probabilities." Let's also call the term $P_{xc}(t)$ the "total probability" of x working nodes, disregarding the connectivity. Hence, if the total probability $P_{2c}(t)$ of state 2 in Figure 8b is multiplied by the connection probability 2/3, we get $P_2(t)$ for state 2 in Figure 8a. Similarly, $P_{2c}(t) \cdot 1/3$ gives the probability of state 1:1. A general expression of this idea for an n-cube is

$$P_j(t) = \sum_{x=j}^{2^n} P_{xc}(t) \cdot P\,(connectivity = j)$$

The first term is the probability that a set of x nodes are working. The second term is the connection probability. x is summed from j to 2^n because of the j-connection requirement. We can then use Equation 4 to obtain the task-based availability for an system with I connected nodes.

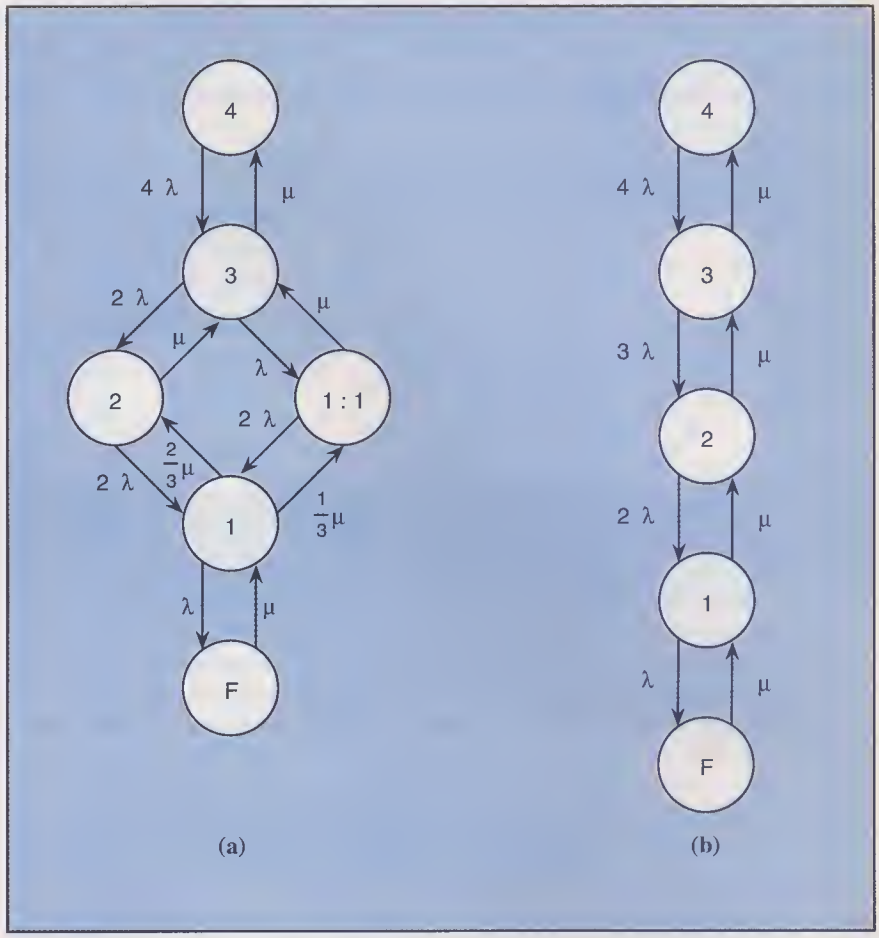


Figure 8. An exact Markov chain of a 2-cube (a); a machine repairman model (b).

Performance-related dependability models

A performance-related dependability measure combines a system's performance and dependability aspects. This measure provides more meaningful information than is available from just a performance or a dependability model, particularly for gracefully degrading systems.

Such a model involves assigning some kind of performance index or reward to a system's various working configurations. Using the notation of the continuous-time Markov chain model, let's represent a generic multiprocessor as a finite-state process $Z(t)$, with $t \geq 0$ and with state space 0, 1, . . . , n. State 0 represents the system's failed state and states 1 through n denote working configurations. The Markov chain could be cyclic or acyclic depending on whether the system is repairable or not. Solving this Markov chain would yield a column vector $\mathbf{P}(t) = p_1(t), p_2(t), \ldots, p_n(t)$, from which we get the probability that the system is in different working states at time t. For example, $p_i(t)$ is the probability that the system is in state i at time t.

Now let's associate a performance index/reward rate r_i with working state i, for $1 \leq i \leq n$. If the system stays in state i for some duration τ, then the reward accumulated in state i is $\tau \cdot r_i$. The system's accumulated reward ($Y(t)$) in time t would be[7]

$$Y(t) = \int_0^t x(\tau) d\tau$$

where $x(\tau)$ is the multiprocessor's reward rate at time τ. In turn, $x(\tau) = \mathbf{r}Z(\tau)$, where $\mathbf{r}$ is an associated reward vector. We can define $\mathbf{r}$ with parameters such as bandwidth, instruction execution rate, or other relevant performance metrics.

Performance-related dependability studies have produced a number of interesting measures. In one of the first attempts, Beaudry introduced measures such as computation reliability and computation availability for degradable multiprocessors.[11] Using the concept of working state probability $p_i(t)$ and an associated reward rate r_i, Beaudry expresses a system's expected computing power at time t as

$$E[x(t)] = \sum_{i=1}^{n} p_i(t) \cdot r_i \qquad (5)$$

Beaudry's model does not consider any specific reward measure or network degradation.

Das and Bhuyan have introduced a parameter called bandwidth availability, $BA(t)$, for shared-memory multiprocessors. This is defined as the expected bandwidth of a multiprocessor at instant t.[8] Bandwidth is a performance measure for synchronous systems, defined as the average number of processor-memory pairs connected in a

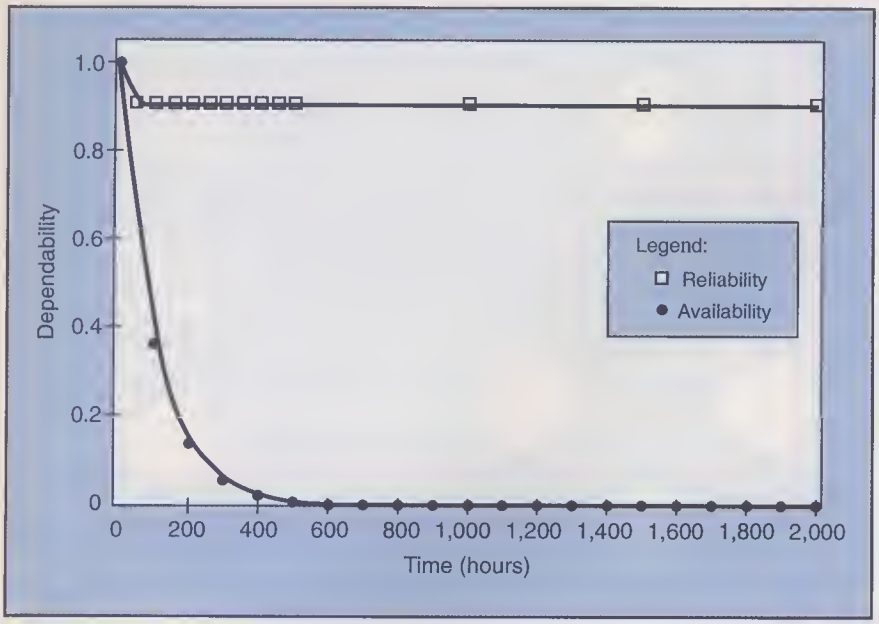


Figure 9. Task-based dependability of a 10-cube allowing 25 percent degradation (at least 768 nodes must be connected). Node failure rate $\lambda = 0.0002$; coverage $c = 0.95$ (a fault is detected and the system correctly reconfigures 95 percent of the time); repair rate $\mu = 0.1$ (repairs are assumed to be perfect). Failure and repair rates are per hour.

cycle. Using a task-based reliability expression (such as Equation 3) for a shared-memory multiprocessor that needs at least I processing elements and J memory modules, bandwidth availability is

$$BA(t) = \sum_{i=I}^{M} \sum_{j=J}^{N} P_{i,j}(t) \cdot BW_{i,j} \qquad (6)$$

where $P_{i,j}(t)$ is the probability that the system is working with i processing elements and j memory modules. $BW_{i,j}$ is the bandwidth (reward r_i in Equation 5) of this $i \times j$ working configuration.

We can also calculate the expected accumulated performance at time t, $E[Y(t)]$, which is the total amount of useful work (average value) that the multiprocessor can perform. We obtain this value by integrating Equation 5 over the observed time frame:

$$E[Y(t)] = \sum_{i=1}^{n} r_i \cdot \int_0^t p_i(\tau) d\tau \qquad (7)$$

Another interesting parameter that has received wide attention recently is performability, introduced by Meyer.[12] Performability is a probability distribution function of accumulative system performance. It gives the probability that a system will complete a given amount of work x in time t. Using the cumulative reward function $Y(t)$, performability is

$$y(x,t) = P[Y(t) \geq x] \qquad (8)$$

Computing this expression is quite complex. For a system with n working states, each state reward distribution involves an integration. (Recent articles on performability contain detailed interpretation of this equation and solution methods.[12,13])

Meyer has worked on various aspects of performability since introducing it a decade ago. He uses a numerical technique for the time-domain convolution equations that predict performability in nonrepairable systems. Recently, he has developed a software package called Metasan[3] that represents a stochastic system as a petri net and determines performability using analysis or simulation. Other researchers have also sought closed-form solutions for different classes of nonrepairable systems in an effort to reduce the computational complexity of finding the probability distribution.

Performability evaluation of multiprocessors has been limited because task-based dependability models are not well developed for some networks and determining the distribution is complex for a system with a large number of states. There has been limited research on crossbar, multiple-bus, and MIN multiprocessors, mainly using bandwidth as the performance measure of the system.

Crossbar systems. Smith and Trivedi[7] have analyzed the performability of the crossbar-based C.mmp multiprocessor using bandwidth as a performance measure. They developed a Markov chain of the 16×16 system, modeling the entire crossbar as a single switch. The distribution is obtained numerically by using a double Laplace transform on variables x and t in Equation 8.

Grassi, Donatiello, and Iazeolla[13] have proposed a closed-form expression for determining the performability distribution of nonrepairable systems. They compute the bandwidth distribution of a crossbar with an ideal (fault-free) network.

Bus-based systems. The double Laplace transform technique has also been used to model the bandwidth distribution of an $M \times N \times B$ multiple-bus architecture[7] by associating each working multiprocessor configuration with its bandwidth. For nonrepairable systems, closed-form expressions for bandwidth distribution can be derived as suggested by Grassi, Donatiello, and Iazeolla.[13]

MIN systems. Based on the idea illustrated in Equation 6, Blake, Reibman, and Trivedi[14] have modeled the expected bandwidth (bandwidth availability) of a 16×16 multiprocessor using an Omega-type MIN. Their model finds $P_{i,j}(t)$, the probability of i processing elements and j memory modules working in a MIN. They multiply this probability by $BW_{i,j}$, the bandwidth of an $i \times j$ MIN subsystem. The model is restrictive in that it cannot be extended to higher order systems, mainly due to the difficulty of finding $P_{i,j}(t)$ and the bandwidth of a randomly truncated MIN with i processing elements and j memory modules.

Numerical examples

The analytical techniques presented above can be used to quickly assess one or more characteristics during a system's life cycle. They can also be used to compare parallel architectures. The following examples illustrating the models' usefulness are confined to task-based analysis, which is the most appropriate measure for multiprocessors.

Quantifying dependability. The models can predict each architecture's reliability or availability at a given time t or predict if the architecture can provide a given

dependability x over a specified time interval. Also, analyzing how sensitive system dependability is to various component failure rates can identify system bottlenecks.[14] Consider a hypercube machine with 1,024 nodes (10-cube). Figure 9 shows the system's reliability and availability, allowing for 25 percent system degradation (I connected), using the hypercube models described above. Note that reliability deteriorates, while availability attains a steady-state value of 0.91.

Comparing dependability. We mentioned earlier that crossbar and shared-bus interconnections, unlike MINs, are not suitable for large multiprocessors. Therefore, any dependability comparisons among multiprocessors using these three types of interconnections should be restricted to small systems.

Consider the design of a 16×16 multiprocessor. Crossbar, shared-bus, and MIN approaches are all feasible and must be considered. Figure 10 shows the system reliability for these three types of systems, assuming they have the same total network complexity (that is, the same failure rate). All three systems operate with approximately the same reliability when degradation is not allowed ($I = J = 16$). However, they have distinctly different reliabilities if we allow for 25 percent degradation ($I = J = 12$). The shared-bus system performs best because it has more alternate paths (eight) than the other designs. The reliability difference between crossbar and MIN systems becomes particularly noticeable when there is system degradation, due to their differing connection capabilities. A switch failure in a Butterfly network disconnects four processor-memory pairs, whereas in a crossbar only one pair is disconnected per failure. On the other hand, a MIN's total network complexity is less, since it has fewer switches. Thus, we can improve a MIN's reliability by adding extra stages of switches.

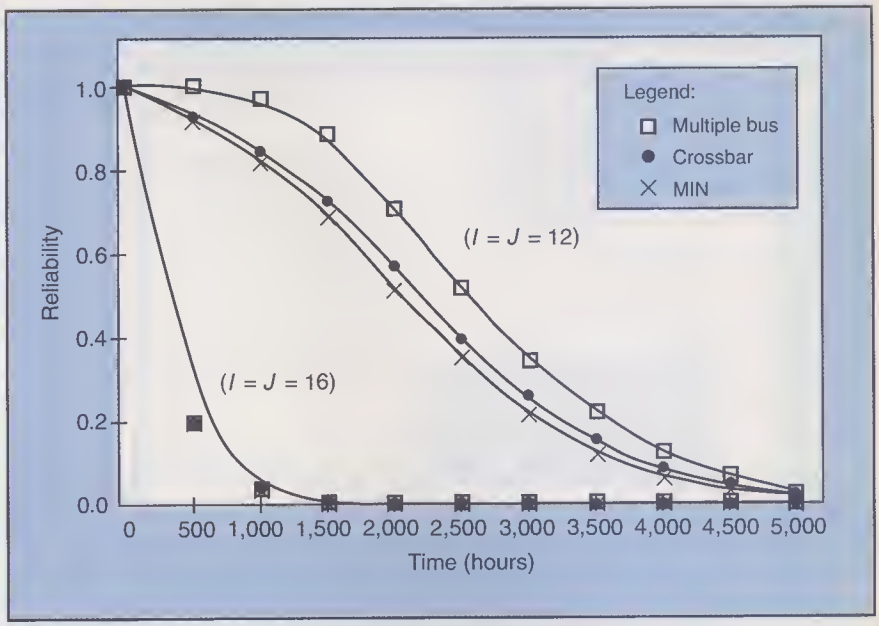


Figure 10. Reliability comparison of 16×16 crossbar, 16×16×8 multiple-bus, and 16×16 Butterfly (MIN) configurations. Processor failure rate $\lambda_p = 0.0001$; memory failure rate $\lambda_m = 0.0001$; crossbar switch failure rate $\lambda_{SE} = 0.000000625$; bus failure rate $\lambda_b = 0.00002$; MIN 4×4 switch failure rate $\lambda_{SE} = 0.00002$.

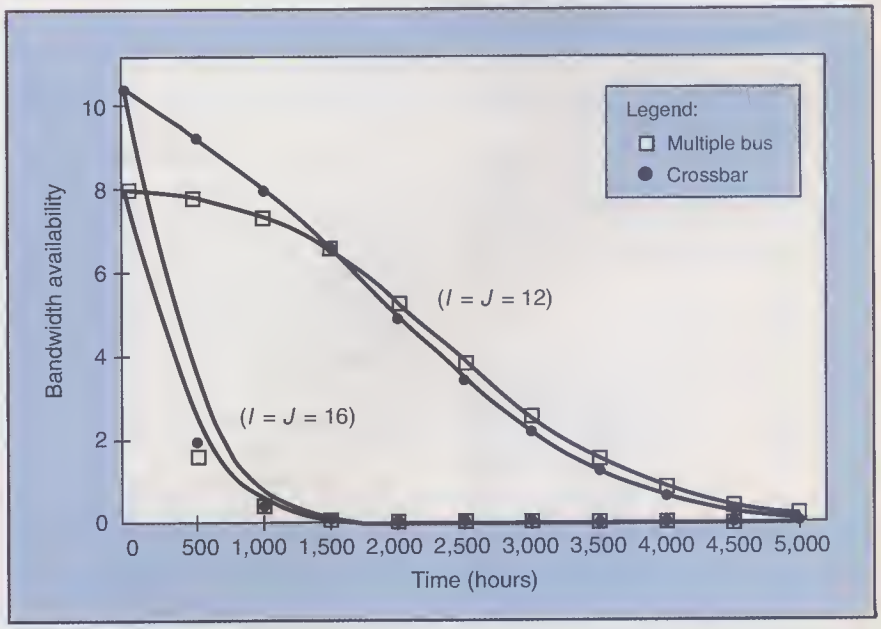


Figure 11. Bandwidth availability comparison of 16×16 crossbar and 16×16×8 multiple-bus architectures. The curves show performance under two different task-based requirements, $I = J = 16$ and $I = J = 12$ (allowing 25 percent degradation). Failure rates: $\lambda_p = \lambda_m = 0.0001$; $\lambda_{SE(crossbar)} = 0.000000625$; $\lambda_{bus} = 0.00002$; coverage c = 1.0.

Quantifying performance-related dependability. Because system performance gradually degrades due to component failure, we must be able to quantify the expected computing power (performance) of a system at any time t. We can do this by replacing the performance index r_i in Equation 5 with a suitable measure.

Suppose we are interested in measuring bandwidth variation of an $M \times N$ synchronous crossbar system. For a uniform-reference model (that is, each processor is equally likely to access each memory), the bandwidth is $BW_{MN} = N \cdot (1 - (1 - 1/N)^M)$.[2] We can then use the bandwidth-availability expression (Equation 6) to compute the expected bandwidth by replacing $P_{i,j}(t)$ with its equivalent from the crossbar state probability expression (Equation 2c) and replacing $BW_{i,j}$ with the crossbar bandwidth just defined. We can find $BA(t)$ for a multiple-bus system in a similar manner. Figure 11 shows the bandwidth availability of

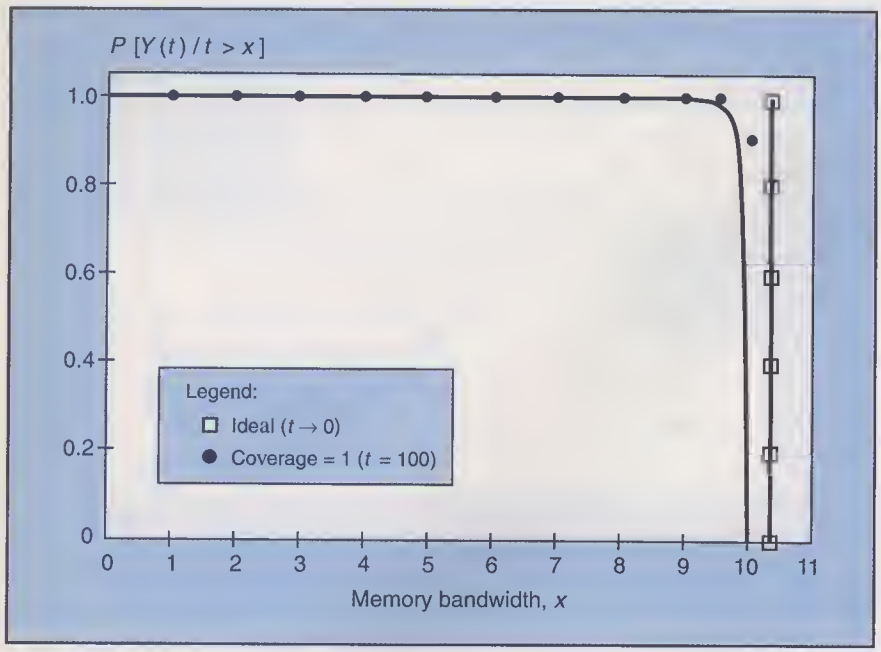


Figure 12. Performability (bandwidth) distribution of a 16×16 crossbar.[7] Failure rates: $\lambda_p = 0.0000689$; $\lambda_m = 0.0002241$; $\lambda_{crossbar} = 0.0002034$; coverage $c = 1.0$.

a 16×16 crossbar and a 16×16×8 multiple-bus system. Although the multiple-bus system initially has less bandwidth, it outperforms the crossbar over time.

Now consider the performability distribution in Figure 12,[7] which shows the time-averaged bandwidth distribution of the 16×16 C.mmp crossbar architecture. The x axis represents the bandwidth (reward x), and the y axis denotes the distribution probability averaged over a time span t, $P[Y(t)/t > x]$. Ideally, as $t \to 0$, the crossbar provides its maximum bandwidth of 10.3 (the vertical line). More realistically, the probability distribution curve reaches zero for a nonzero time span when the bandwidth x increases. For example, the probability of attaining a bandwidth of 10 for a mission time of 100 hours is less than 1 (about 0.9) due to component failure. This type of information is important when assessing an architecture's suitability for a specified task.

Software packages

Many software tools have been developed for analyzing the dependability of complex systems. (Johnson and Malek have written a detailed survey of these tools.[3]) The tools discussed below are a good sample of those available and cover most modeling techniques and objectives.

The Automated Reliability Interactive Estimation System (ARIES) was one of the first tools for predicting the reliability of repairable and nonrepairable systems. It uses homogeneous Markov chain models for determining reliability, mean time to first failure, and the reliability improvement factor. It cannot model general failure distribution.

NASA Langley Research Center and Raytheon have developed a software package called Computer-Aided Reliability Estimation III (CARE III) to model highly reliable nonrepairable systems. It accepts fault trees, nonhomogeneous Markov chains, and semi-Markov chains as input. The model provides numerical values or graphical plots of reliability and coverage data. It cannot model repairable behavior.

The Hybrid Automated Reliability Predictor (HARP) was developed at Duke University and is now available through NASA Langley. It can compute the reliability and instantaneous availability of nonrepairable and repairable systems, respectively. HARP implements a fault occurrence and repair model that uses Markov chains with multiple distributions, and a fault- and error-handling model with seven types of analytical modeling techniques. It converts a fault tree to a Markov chain or accepts a Markov chain representation as input. It cannot compute steady-state availability or mean time between failures.

The System Availability Estimator (Save), developed at IBM, evaluates the reliability and availability of nonrepairable and repairable systems. It uses a homogeneous Markov chain model to calculate steady-state availability, sensitivity, and mean time to failure. Save accepts system descriptions in a high-level language and solves the model using either a Markov or Monte Carlo simulation approach. The model is restricted to exponential distribution.

The Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE), being developed at Duke University, is intended to provide a hybrid and hierarchical way to evaluate the reliability and availability of nonrepairable and repairable systems. It is also intended for use in evaluating performance-related dependability. This tool will accept reliability block diagrams, fault trees, or Markov chains as input for predicting reliability and both transient and steady-state availability.

While these evaluation tools are sophisticated enough to analyze complex systems, most of them require a reliability block diagram, fault tree, or Markov chain as input, which limits their usefulness for large multiprocessor architectures. Unless the dependability model is very simple, it is extremely difficult to develop any kind of input model for a large multiprocessor. When the solution involves a Markov approach, the size and nature of a single-level Markov chain makes these tools particularly inadequate. A hierarchical approach (divide and conquer) seems necessary to evaluate multiprocessors with these tools. Such an approach would divide a system into several subsystems, with each subsystem evaluated independently before the complete system is examined.

Although we have focused on four types of architectures (due to their commercial availability), a number of other topologies for parallel-system design have been proposed.

As more powerful systems evolve, their evaluation will depend largely on the organization of their interconnection networks. Each such network must be addressed by an appropriate model. We expect that the practicality of task-based evaluation will make it the metric of choice. Future shared-memory systems could use an approach like the MIN modeling techniques discussed by Das, Tien, and Bhuyan,[9] and distributed memory systems could build

on the hypercube model developed by Das and Kim.[10]

The modeling and evaluation of parallel computers is an evolving research area. Most models address only hardware failure. A more realistic evaluation requires the inclusion of software faults and different types of transient and intermittent faults. Performance-related dependability evaluation of parallel computers is another active research area. We need more meaningful, as well as powerful, models for specific multiprocessor architectures. In turn, these should motivate researchers to develop more sophisticated software tools to aid in analysis. ■

References

1. J.C. Laprie and A. Coste, "Dependability: A Unifying Concept for Reliable Computing," *Proc. 12th Int'l Symp. Fault-Tolerant Computing*, 1982, Computer Society Press, Los Alamitos, Calif., Order No. 418 (microfiche only), pp. 18-21.

2. C.L. Wu and T.Y. Feng, *Interconnection Networks for Parallel and Distributed Processing*, Computer Society Press, Los Alamitos, Calif., Order No. 574, 1984.

3. A.M. Johnson and M. Malek, "Survey of Software Tools for Evaluating Reliability, Availability, and Serviceability," *ACM Computing Surveys*, Vol. 20, No. 4, Dec. 1988, pp. 227-269.

4. *IEEE Trans. Reliability*, special issue on reliability of parallel and distributed computing networks, Vol. 38, No. 1, Apr. 1989.

5. A. Grnarov and M. Gerla, "Multiterminal Reliability Analysis of Distributed Processing Systems," *Proc. Int'l Conf. Parallel Processing*, 1981, Computer Society Press, Los Alamitos, Calif., Order No. 354 (microfiche only), pp. 79-86.

6. A.D. Ingle and D.P. Siewiorek, "Reliability Models for Multiprocessor Systems With and Without Periodic Maintenance," *Proc. 7th Int'l Symp. Fault-Tolerant Computing*, 1977, Computer Society Press, Los Alamitos, Calif., Order No. 156 (microfiche only), pp. 3-9.

7. R.M. Smith and K.S. Trivedi, "Performability Analysis of Two-Multiprocessor Systems," *Proc. 17th Int'l Symp. Fault-Tolerant Computing*, 1988, Computer Society Press, Los Alamitos, Calif., Order No. 778 (microfiche only), pp. 224-229.

8. C.R. Das and L.N. Bhuyan, "Bandwidth Availability of Multiple-bus Multiprocessors," *IEEE Trans. Computers*, Vol C-34, Oct. 1985, pp. 918-926.

9. C.R. Das, L. Tien, and L.N. Bhuyan, "Availability Evaluation of MIN-Connected Multiprocessors Using Decomposition Technique," *Proc. 20th Int'l Symp. Fault-Tolerant Computing*, 1990 pp. 176-183.

10. C.R. Das and J. Kim, "An Analytical Model for Computing Hypercube Availability," *Proc. 19th Int'l Symp. Fault-Tolerant Computing*, 1989, Computer Society Press, Los Alamitos, Calif., Order No. 1959, pp. 530-537.

11. M.D. Beaudry, "Performance-Related Reliability Measures for Computing Systems," *IEEE Trans. Computers*, Vol C-27, Jan. 1978, pp. 540-547.

12. J.F. Meyer, "On Evaluating the Performability of Degradable Computing Systems," *IEEE Trans. Computers*, Vol C-29, Aug. 1980, pp. 720-731.

13. V. Grassi, L. Donatiello, and G. Iazeolla, "Performability Evaluation of Multicomponent Fault-Tolerant Systems," *IEEE Trans. Reliability*, Vol. 37, June 1988, pp. 216-222.

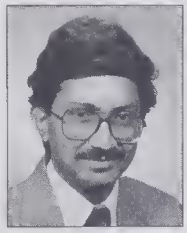
14. J.T. Blake, A.L. Reibman, and K.S. Trivedi, "Sensitivity Analysis of Reliability and Performability Measures for Multiprocessor Systems," *Proc. ACM Conf. Measurement and Modeling of Computing Systems*, 1988, pp. 177-186.

Acknowledgments

This research was supported in part by the National Science Foundation under grants CCR-8810131, MIP-8710753, and MIP-8807761. We would like to thank the referees for their comments, which contributed to the quality of this article.

Chita R. Das is an assistant professor in the Electrical and Computer Engineering Department at Pennsylvania State University. His research interests are in parallel/distributed computing, performance and dependability evaluation, and fault-tolerant computing. He received his MS degree in electrical engineering from Regional Engineering College, Rourkela, India, and his PhD degree in computer science from the University of Southwestern Louisiana in 1986. He is a member of the IEEE Computer Society, the IEEE, and the ACM.

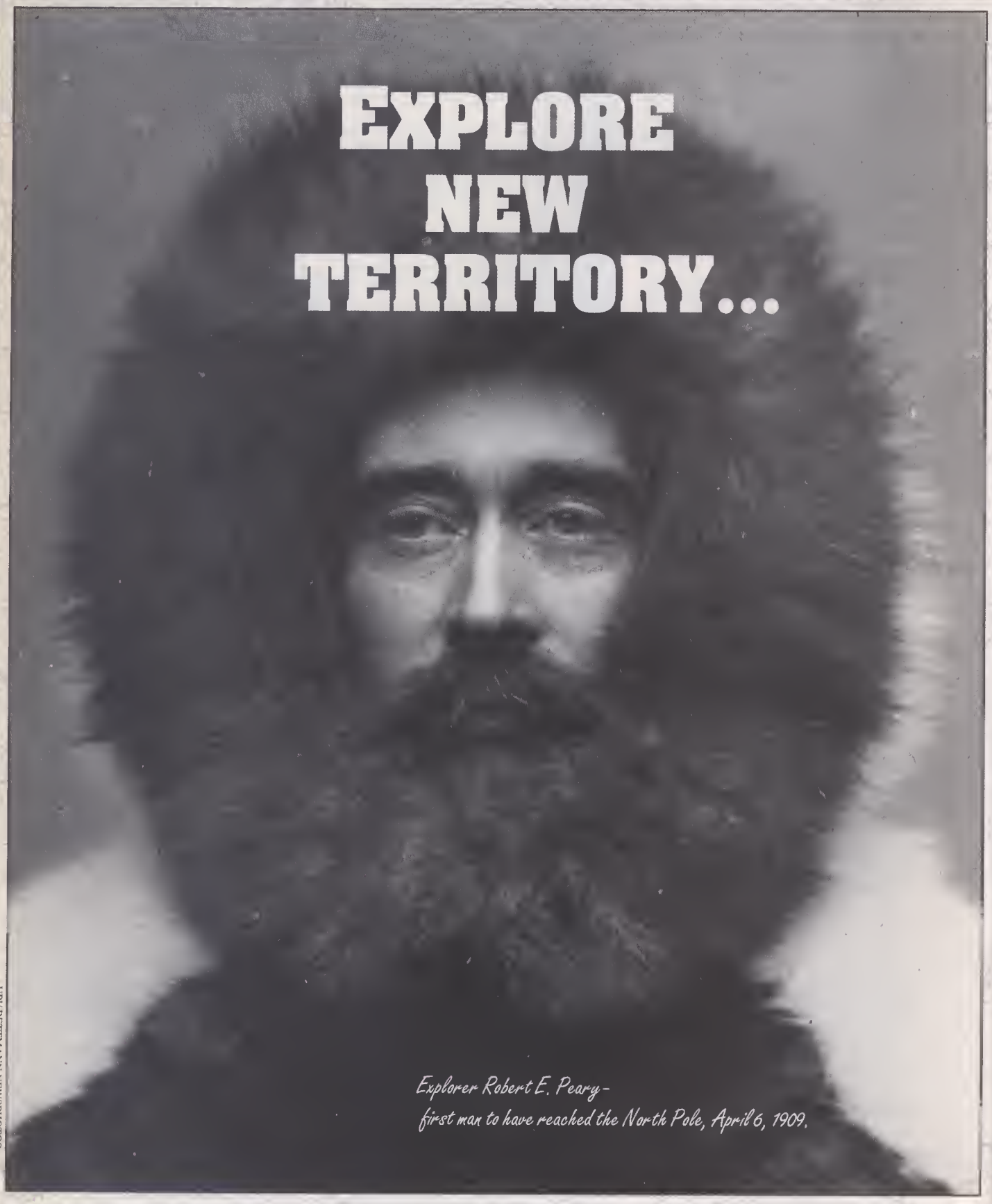
Jeffrey T. Kreulen is working toward a PhD in computer engineering at Pennsylvania State University. His interests include parallel processing, computer architecture, and programming languages. He received a BS degree in applied mathematics (computer science) from Carnegie Mellon University in 1986, and an MS degree in electrical engineering from Pennsylvania State University in 1988. He is a student member of the IEEE Computer Society, the IEEE, and the ACM.



Matthew J. Thazhuthaveetil is an assistant professor of computer engineering at Pennsylvania State University. His research interest is in high-performance computer architecture. He received the BTech degree in electrical engineering from the Indian Institute of Technology, Madras, in 1980, and the MS and PhD degrees in computer sciences from the University of Wisconsin, Madison, in 1983 and 1986, respectively. He is a member of the IEEE and the ACM.

Laxmi N. Bhuyan is an associate professor in computer science at Texas A&M University. His research interests include parallel and distributed computer architecture, fault-tolerant computing, and performance evaluation. Bhuyan received BS and MS degrees in electrical engineering from Regional Engineering College, Rourkela, India. He received a PhD in computer engineering from Wayne State University, Detroit, in 1982. Bhuyan is a senior member of the IEEE and a distinguished visitor of the IEEE Computer Society.

Readers can contact Das at the Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802.



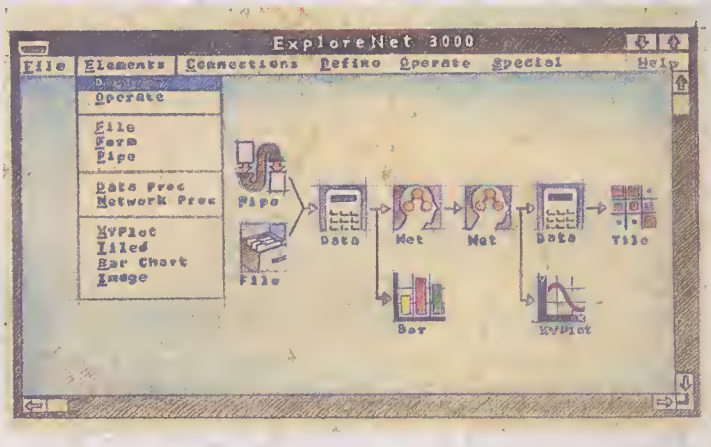
UPI/ BETTMANN NEWSPHOTOS

Microsoft Windows and Excel are trademarks of Microsoft Corp. Lotus 1-2-3 is a trademark of Lotus Development Corp. dBase II is a trademark of Ashton-Tate.

Find Hidden Answers.

Order Now
Save over $1000 on Training and Support

EXPLORENET 3000™ is advanced neural network software for the PC that helps you find hidden answers in your data. You will experience the ease of developing your application within Microsoft Windows™ 3.0 environment (compatible with over 3,000 other programs on the PC/AT). And best of all, you can develop applications without programming! ExploreNet 3000 applications are 100% compatible with HNC's ANZA Plus™ coprocessors.



File Module

Support fixed and variable length ASCII and binary files, which can be generated by 99% of all programs. For instance, output from such popular programs as Lotus 1-2-3™, dBase II™ and Excel™ can be used without modification.

Data Transformation Module

Perform pre-and post-processing operations by combining functions from a library of over 50 routines. Supports scalar, vector and matrix operations.

Pipe Module

Receive input from and output to thousands of other programs including expert systems, databases and applications programs running on the PC under MS-Windows.

Display Modules

Watch your network train and learn by monitoring the results using six different display options: Plots, Tiles, Images, Forms and Charts (pie charts and bar charts).

Network Module

Tailor any of the 18 well-known neural networks to your application, or modify the architecture of one of the three experimental networks to your specifications.

ExploreNet 3000 will get you up and running faster than any other neural network product on the market—we'll back it with a 30-day money-back guarantee.

Call 1-800-HNC-EXPR

HNC™

5501 Oberlin Drive
San Diego, CA 92121
(619) 546-8877
Fax (619) 452-6524

Reader Service Number 4

CALL FOR PAPERS

28TH ACM/IEEE DESIGN AUTOMATION CONFERENCE®

SAN FRANCISCO MOSCONE CENTER • JUNE 17-21, 1991

DAC is the premier conference devoted solely to the field of Design Automation. All aspects of the use of computers as aids to the design process, from conceptual design through to manufacturing, are included. Four session types are included: regular paper sessions, short paper sessions, panels, and tutorials.

REQUIREMENTS FOR SUBMISSION OF PAPERS

Authors should submit their papers to the Program Chair no later than November 7, 1990. Each submission should include one cover page and eight stapled copies of the complete manuscript.

The one cover page should include:

- Name, affiliation, and complete address for each author
- A designated contact person including fax (if available), and telephone number
- A designated presenter, should the paper be accepted
- The following signed statement: "All appropriate organizational approvals for the publication of this paper have been obtained. If accepted, the author(s) will prepare the final manuscript in time for inclusion in the Conference proceedings and will present the paper at the Conference."

The eight copies of the complete manuscript should each include:

- Title of paper, 60 word abstract indicating significance of contribution, and a list of category numbers (given below), **ordered by relevancy,** most closely matching the content of the paper. **To permit a blind review, do not include name(s) of author(s).**
- The complete text of the paper in English, including all illustrations and references, not exceeding 5000 words. The length of the paper is considered in the review process.

Notice of acceptance will be mailed to the contact person by February 15, 1991. Authors of accepted papers must sign a copyright release form.

TOPICS OF INTEREST

Authors are invited to submit original technical papers describing recent and novel research or engineering developments in all areas of Design Automation. Topics include, but are not limited to:

1.1 Electrical Simulation
1.2 Discrete Simulation
1.3 Timing Verification
2.1 Test Validation and Testability Analysis
2.2 Test Pattern Generation and Design-for-Test
2.3 Formal Methods for Design Verification
3.1 Floorplanning and Placement
3.2 Global and Detailed Routing
3.3 Physical Module Generation
3.4 Symbolic Layout and Compaction
3.5 Complete Layout Systems
3.6 Layout Verification (DRC, ERC)
4.1 Logic Synthesis and Optimization
4.2 Register-Level Synthesis and Optimization
4.3 Behavioral Synthesis and System-Level Design Aids
5.1 Behavioral and Hardware Description Languages
5.2 Design Systems and Databases
6.1 DA for IC Fabrication
6.2 Computer Aids for Manufacturing
7.1 DA for Analog Circuits
7.2 High Speed Systems and Microwave DA
7.3 DA for Electronic Packaging
8.1 Human Factors in DA
8.2 Frameworks and Software Engineering in DA

PANELS, TUTORIALS, SPECIAL TOPIC SESSIONS

Proposals for topics of Panels, Tutorial Sessions, and Full-Day Tutorials should be submitted to the Program Chair no later than November 7, 1990. Each proposal should not exceed two pages in length and should include a description of the topic, structure of the session or tutorial, and a list of participants. Contact should be made with all participants in advance.

Special Topic Sessions may be either independent papers with a common theme or a set of closely related papers describing an overall system. In both cases, independent reviews of each paper and evaluation of the session as a whole will be used to select sessions. Proposals for Special Topic Sessions should be submitted along with the papers to be included in the session and should describe the theme of the session. These proposals and papers must be submitted by November 7, 1990 to allow for a complete review.

Birds of a Feather meetings are informal groups to discuss technical items of common interest. These very informal noncommercial meetings are held after normal Conference hours. Submissions should be one paragraph in length.

PROGRAM CHAIR

MP Associates, Inc.
ATTN: Alfred E. Dunlop
Program Chair, 28th DAC
7490 Clubhouse Rd., Suite 102
Boulder, Colorado, 80301
For information call: (303) 530-4333

sponsored by acm SIGDA® IEEE COMPUTER SOCIETY DATC® THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC. in cooperation with IEEE CIRCUITS & SYSTEMS SOCIETY®

A System Architecture for Fault Tolerance in Concurrent Software

Massimo Ancona, Gabriella Dodero, and Vittoria Gianuzzi, University of Genova

Andrea Clematis, Italian National Research Council

Eduardo B. Fernandez, Florida Atlantic University

Robotics, process control, navigational systems, and other critical computer applications demand reliable software. Such systems generally comprise a set of concurrent, cooperating processes.

Several researchers have developed methods and tools that use redundancy to help critical systems tolerate errors caused by software faults. Two of the most widely used techniques for sequential software are recovery block programming[1] and *N*-version programming.[2] The most suitable mechanisms for concurrent systems are programmer transparent coordination[3] and conversation.[1] All four of these approaches are general and apply to any type of computation. We can consider them different types of design diversity. (The sidebar titled "Software fault-tolerance techniques" details the approaches.)

Unfortunately, the few languages that provide adequate syntax and runtime support to implement fault-tolerant mechanisms are still experimental. Usually, the programmer must describe the desired fault-tolerance policy explicitly, greatly complicating the program's implementation, readability, and maintenance.

Programmers need an environment that effectively supports the development of fault-tolerant programs. At the very least, such a system must let the user

Our proposed architecture separates the application from the recovery software, giving programmers a single environment that lets them use the most appropriate fault-tolerance scheme.

- select and apply the most suitable fault-tolerant mechanism for the application program;
- modify that mechanism as required by the program or to experiment with new fault-tolerance schemes; and
- modify and maintain the application program without interfering much with the recovery structure.

Our proposed system architecture fulfills these requirements and can support a variety of policies in a structured way. It treats the application program and the recovery software — called the *Recovery Metaprogram* — separately. The RMP acts like a programmer who monitors and (possibly) modifies the execution of the application program. (The sidebar titled "Other similar approaches" discusses how other researchers are handling fault tolerance in related ways.)

To simplify our presentation of the RMP approach, we will assume that the fault model is limited to faults originating in the application software, and that the hardware and kernel layers can mask their own faults from the RMP. Also, we will not consider relationships between backward and forward error recovery. Exceptions at other layers will be handled within those layers; they will not be signalled to the RMP, and no other exceptions will be raised within the RMP.

Recovery architecture

Our software fault-tolerance architecture contains three main components: the application component, the recovery component, and the kernel.

The application component is the user's

0018-9162/90/1000-0023$01.00 © 1990 IEEE

view of the computational system, for example, as a set of communicating processes. In the recovery component, the RMP coordinates the application program's execution. The RMP is basically a set of processes that implement fault-tolerance mechanisms. The kernel implements an ordinary multiprocessing kernel augmented by a set of fault-tolerance primitives, which the RMP invokes as kernel requests. The kernel also maintains data structures relevant to process management; RMP calls can update some portions of these structures. This organization can support any fault-tolerance policy specified in the RMP.

Figure 1 illustrates the relationship between the application program and the RMP and shows how the RMP superimposes the recovery block control structure on an application program. The kernel is the medium between them.

In practice, the RMP monitors the application program much as a programmer observes software through a debugger. The RMP inserts a number of breakpoints in the program. When one of them is reached, the application program is suspended and the kernel activates the RMP, which performs actions to support the chosen fault-toler-

Software fault-tolerance techniques

Sequential

The recovery block construct is based on checkpointing and redundancy. Several alternate try blocks, all implementing the same functionality, combine to achieve one fault-tolerant execution. At block entry, the program status is saved and the first alternate is executed. The computed results are checked before block exit. If this acceptance test fails, the computation is rolled back to block entry status and an alternate try block is executed.

Assume that two alternate functions, SqrtA and SqrtB, both implement the square root function. The syntax Randell[1] proposes for such a recovery block is then

```
ensure      |x² – y| < tolerable
by          x := SqrtA(y)
else by     x := SqrtB(y)
else error
```

N-version programming uses redundancy to mask software faults. Several different versions of the same software execute, and some hardware or software unit compares their computed results. The majority result is accepted. To achieve a fast response time, the different versions usually execute in parallel on different machines. Checkpointing and rollback are not required in this case, but different initial status copies must be created.

Concurrent

Applying recovery block or *N*-version programming schemes without coordination in concurrent programs can incur a domino effect,[1] since errors can be exported to other processes via communication. For example, a fault requiring a rollback in one process can provoke a rollback in another process because of communication between them. This, in turn, could affect the whole set of processes, causing the computation to restart from its initial state. Thus, recovery block or *N*-version programming mechanisms must be extended to apply to concurrent environments.

The conversation scheme is essentially a concurrent extension of the recovery block scheme in which a group of processes can interact safely. Each process entering a conversation is checkpointed. To prevent information smuggling (that is, spreading data that has not been validated),[4] the process can then communicate only with processes that are already in the conversation. When all processes have completed their operations, a global acceptance test occurs. If it fails, all processes roll back and execute their next alternates; if it is satisfied, all processes discard their checkpoints and exit synchronously from the conversation.

The conversation scheme guarantees that no subsequent error can roll a process back to a point before a successful acceptance test; hence, no domino effect can occur. Several implementations have been proposed in the literature to extend and adapt this scheme to various environments and programming styles.

Let's assume that conversation C1 involves *m* processes, $P_1, \ldots, P_m$, and that the global acceptance test is the logical AND of the various processes' acceptance tests. The portion of P_1 involved in C1 can be specified, for example, as a *name-linked* recovery block,[4] that is, a recovery block with *n* alternates and with C1 as a prefix:

```
Process P1 = . . .
        \\* definition of procedures
        P1C1₁, P1C1₂, . . . P1C1ₙ, and
        error, and of the Boolean
        function ATP1C1 *\\
begin . . .
        Conv     C1
        ensure   ATP1C1
                 by        P1C1₁
                 else by   P1C1₂
                 . . .
                 else by   P1C1ₙ
                 else error
        end Conv
```

An alternative mechanism for defining a group of consistent recovery points is programmer transparent coordination. PTC relies on an intelligent processor, which prevents the domino effect by automatically taking additional recovery points within recovery blocks defined in a process. Therefore, a programmer can use recovery blocks freely in each process without worrying about coordinating checkpoints and rollbacks in other processes.

PTC differs from conversation in that the programmer must explicitly state conversation boundaries, while PTC is a transparent mechanism. Also, the conversation acceptance test is synchronous, while PTC lets processes proceed independently as long as a process executing a recovery block keeps the corresponding checkpoints until all related recovery blocks in other processes have succeeded.

ance mechanism. The RMP is then suspended again, and the application program is reactivated until the next breakpoint is reached.

The application program. Our system lets users limit their intervention in the application software to indicating which portions of the program are involved in fault tolerance. Control-flow specification is left to the RMP. For instance, programmers can use labels to specify recovery points, and some Boolean functions can be chosen as the validation tests.

Such a scheme requires a programming tool to support the interface between the application and the RMP. This tool must collect information on the portions of the application program to be monitored by the RMP as well as implement the program code to allow control transfers, as shown in Figure 1.

We will now use two examples — the recovery block and the conversation — to detail how an application can specify a mechanism.

The recovery block. Consider what happens if we want to use recovery blocks in the Fortran program in Figure 2. (For ease of reference, we consider alternates to be procedures and, therefore, named entities.)

We cannot insert a special construct in a Fortran program to specify that procedures SqrtA and SqrtB are alternates and that SqrtB should execute only if SqrtA fails the acceptance test (Verify). Thus, the programming environment must collect information about these components, and any links among the procedures must be expressed in the RMP, where a recovery block mechanism is made explicit to activate them. To specify the control flow, the RMP must be able to reference the beginning, end, and other components of the recovery block, such as labels 10 and 20, SqrtA, SqrtB, and Verify.

Conversation. We use the syntax in Figure 3 if there is no construct to define which portions of process P1 participate in conversation C1. If software fault tolerance is not needed, control flow consists simply of the execution of procedure $P1C1_1$. Otherwise, the RMP coordinates execution in a manner similar to the recovery block scheme; for example, $P1C1_2$ is activated if $P1C1_1$ fails.

We can use basically the same RMP both with languages that support dedicated fault-tolerance constructs and with languages that do not support them. In the latter case, a specific tool in the programming environment gives the RMP information about the application.

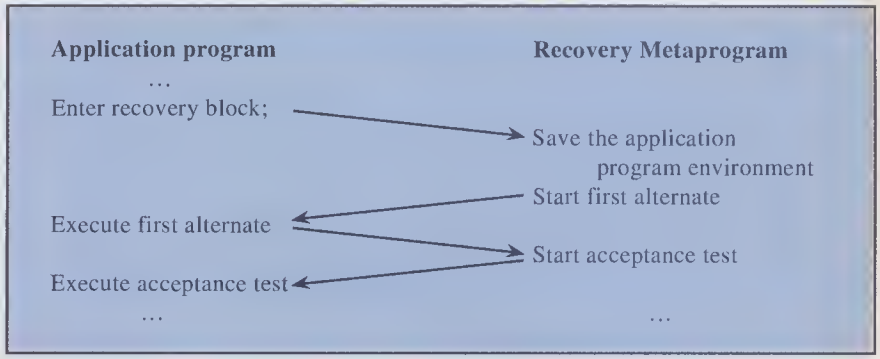


Figure 1. Control flow between the application program and the Recovery Metaprogram.

```
main call:
     . . .
10   SqrtA(X,Y)
20   Continue
     . . .

subroutines:
     Subroutine SqrtA(. . . )          //* 1st alternate *//
     . . .
     End

     Subroutine SqrtB(. . . )          //* 2nd alternate *//
     . . .
     End

     Logical function Verify(. . . )   //* acceptance test *//
     . . .
     End
```

Figure 2. Recovery block example.

The RMP view of the application component. At least the following items must be defined in the application program and made accessible to the RMP:

- The entry points (that is, the beginning and end) of each portion of an application process under RMP control. Constructs denoting entry points vary according to the application language; for instance, labels, pragmas, or separate descriptive units can be used.
- The alternates for an execution path, to be executed in sequence (recovery block) or in parallel (*N*-version programming).
- The validation test, a Boolean function without side effects.

```
Process P1 =
        . . .
        //* definition of procedures
        P1C1_1, P1C1_2, . . . P1C1_n, and
        error, and of the Boolean
        function ATP1C1 *//

begin . . .
   C1init: 1C1_1
   C1end: . . .
```

Figure 3. Conversation example.

Since each of these entities will belong to some process, the RMP must know the set of application processes. The RMP also must know the *environment* of a process, that is, the set of all current valid associa-

Other similar approaches

Crespi and Natali[1] have proposed *superprograms* to extend existing languages with fault-tolerant features. Superprograms operate on application program entities such as processes and data structures, as well as on resources provided by the execution environment. The language for such a structure uses debugging-like commands, some application-like constructs, and hardware configuration descriptions. The RMP is an extension of this supervisory-program concept. (We presented an earlier version of our architecture in a previous work.[2])

Researchers are showing greater interest in unifying various software fault-tolerance techniques. For example, Laprie et al.[3] have proposed a structured definition for architectures offering fault tolerance in both hardware and software. They consider and evaluate specific software fault-tolerance architectures — recovery block, *N*-version programming, and *N*-version self checking — with respect to their reliability. Their global view considers special treatment of specific classes of faults or techniques only when necessary.

Purtilo and Jalote[4] have proposed a system to implement the recovery block and *N*-version programming schemes in multiple programming languages. Their goal was to ensure diversity by letting programmers develop redundant modules in different languages.

References

1. S. Crespi-Reghizzi and A. Natali, "Soft Tools for Adding Fault Tolerance to Concurrent, Distributed Programs," in *MML: Linea di Programmazione per Sistemi Multimicroprocessore*, S. Crespi-Reghizzi and O. D'Antona, eds., Italian National Research Council, 1985.
2. M. Ancona et al., "A System Architecture for Software Fault Tolerance," in *Fault-Tolerant Computing Systems*, Informatik-Fachberichte, No. 147, Springer-Verlag, New York, 1987, pp. 273-284.
3. J.-C. Laprie et al., "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," *Computer*, Vol. 23, No. 7, July 1990, pp. 39-51.
4. J.M. Purtilo and P. Jalote, "A System for Supporting Multilanguage Versions for Software Fault Tolerance," *Proc. 19th Int'l Symp. Fault-Tolerant Computing*, 1989, Computer Society Press, Los Alamitos, Calif., Order No. 1959, pp. 268-274.

tions between variables and values. An environment with no valid associations is a *faulty* environment resulting from a software error.

Environments can constitute checkpoints for a rollback strategy. Depending on the chosen fault-tolerance strategy, the RMP specifies when such checkpoints must be saved, restored, duplicated, discarded, or used in process execution. The kernel implements these operations using either hardware or software resources (for example, stable storage, clone processes, and virtual memory).

To be managed by the RMP, an entity must be one of a set of dedicated data types defined in the RMP implementation language (see Figure 4). With these types, we can classify the entities in Figure 2 as follows: Labels 10 and 20 are Entrypoints, SqrtA and SqrtB are Codesections, and Verify is the Validationtest. We can define the entities in Figure 3 in a similar manner: C1init and C1end are Entrypoints, procedures $P1C1_1$, $P1C1_2$, $P1C1_n$, and error are Codesections, and function ATP1C1 is the Validationtest

We can also construct sets out of processes or environments and handle them with the usual set operators for testing membership, union, cardinality, and the like.

Process — An application process instance (compatible with the process type as managed by the kernel).

Entrypoint — An address in the process code area. When no value is needed, the constant Nil is used.

Codesection — An application process procedure.

Validationtest — A Boolean function with no side effects.

Environment — A representation of a process' current data area. A faulty environment is denoted by the constant Faulty, and an empty (not initialized) environment by Empty. The environment in use at a given moment is Current.

Figure 4. RMP data types.

The kernel interface to the RMP. The kernel provides the operations on entities belonging to the above data types. Figure 5 lists the primitives that invoke these services, using the following syntax: input parameters to kernel calls are in parenthesis, while call results follow a colon.

The Save, Discard, Restore, and Continue functions update process-related information in the kernel and return to the RMP without rescheduling (that is, the application program is not activated).

The Restrict function controls interprocess communication and prevents information smuggling among processes in conversation schemes. Processes usually communicate directly via messages or indirectly via monitors or remote procedure calls. Most proposed conversation schemes require that the kernel controls communication.

Caller and Where_Is are query functions. The Execute, Evaluate, Vote, and Terminate functions all suspend the RMP, activate (or abort) the specified process on the specified procedure after rescheduling, and reactivate the RMP when the procedure has executed (or been forced to terminate).

The parameters in Figure 5 are minimal; more can be added. For instance, processor identification can be inserted if distributed execution is needed.

The RMP structure. We have discussed the RMP data types and a minimal set of kernel primitives, but what about the RMP's language and general structure?

Since the RMP monitors the application program, it is responsible for issuing suitable kernel calls to initialize and subsequently activate the program's processes. The RMP repeatedly invokes the Continue primitive to tell the kernel to set all processes to the ready state and start their processing. The RMP then waits until at least one process reaches an entry point and suspends execution.

Since many RMP actions can execute concurrently, it is preferable — although not mandatory — to express the RMP in a concurrent language. The kernel implements a concurrent virtual machine on top of any type of hardware, but given a distributed or multiprocessor system, the specification of concurrent activities in the RMP could lead to distributed execution as well.

We specify the RMP in the Communicating Sequential Processes[5] language, which provides an elegant and concise notation for expressing RMP actions. We use a slightly extended notation for guards:

at *Entrypoint* value → *Process*

which means the process is activated when the application process is suspended at the specified address. Figure 6 summarizes the other CSP notations.

RMP examples

The following examples use the data structures shown in Figure 7. We omit some of the subscripts when they are clear from the context. All the data structures have values generated by the compiler (or some tool of the programming environment) and are accessed by the RMP as read-only data (with the exception of $CVentered_j$ and variables of the Environment type).

The first statement of the RMP main program activates all processes defined in the application program:

$$\|_{p_i \in Pset} (\text{Continue}(p_i, start_i, Empty, false));$$

After that, a number of RMP processes are started in accordance with the chosen strat-

Save(*Process*): *Environment* — A checkpointing primitive. The current environment of the referenced process is saved and its name returned as a result.

Discard(*Environment*) — The specified environment is discarded.

Restore(*Process*, *Environment*) — The environment previously saved is restored.

Continue(*Process*, *Entrypoint*, *Environment*) — The process' status is set to ready; it will be reactivated from the specified address.

Restrict(*Process*, set of *Processes*, set of *Processes*) — A communications restriction on the specified processes. The first process can communicate only with processes in either of the other two sets. This allows asynchronous entrance into a conversation in three ways:
- Communication to (or from) processes in the first set can be performed immediately.
- Communication to (or from) processes in the second set delays the sending (or receiving) process until a subsequent Restrict operation allows the communication (this is required to implement asynchronous entry).
- Communication to (or from) a process outside either set causes a nonrecoverable error in the sending (or receiving) process (its environment becomes faulty).

Caller(*Process*): *Process* — If the input parameter is a monitor or a remote procedure, the output value is the process for which the monitor is working.

Where_Is(*Process*): *Entrypoint* — If the process is idle or waiting, the returned value is the address at which it is suspended. Otherwise, it is Nil.

Execute(*Process*, *Codesection*, *Environment*): *Environment* — Starts execution of the application process with the indicated code section and environment. Activation returns an environment, which can be faulty if an error has occurred.

Evaluate(*Process*, *Validationtest*, *Environment*): *Boolean* — Returns the Boolean result of the validation test performed on the given environment. If the actual environment parameter is Faulty, the result is "false."

Vote(*Validationtest*, set of *Environments*): *Environment* — The environments listed as input parameters are compared by the given validation test, and the resulting environment is returned. The application program specifies the voting policy in the Validationtest function.

Terminate(*Process*) — The specified process aborts (after a nonrecoverable error).

Figure 5. Kernel primitives.

Notation	Meaning
P, Q	Processes
$(a \rightarrow P \mid b \rightarrow Q)$	a then P choice b then Q (a and b are guards of the two processes P and Q)
$P \parallel Q$	P in parallel with Q
$\|_{1 \le i \le n} P_i$	$P_1 \parallel P_2 \parallel \ldots \parallel P_n$
$*P$	repeat P
$b* P$	while b repeat P

Figure 6. Communicating Sequential Processes (CSP) symbols.

Pset — Set of processes defined in the application program.

p_i — ith process of *Pset*.

$start_i$ — Start address of process p_i.

$entry_{i,j}$ — Entry point of process p_i in the jth controlled block.

$exit_{i,j}$ — Exit point of process p_i in the jth controlled block. $Start_i$, $entry_{i,j}$, and $exit_{i,j}$ are of the *Entrypoint* type.

$CSec_{i,j,k}$ — Procedure defined in process p_i, in the jth controlled block, implcmenting the kth alternate (*Codesection* type).

$VTest_{i,j}$ — Validation test defined in p_i and relative to the jth controlled block (*Validationtest* type).

RBset — Set of recovery blocks defined in the application program.

NVset — Set of *N*-version programming blocks defined in the application program. (Since recovery blocks and *N*-version programming blocks are defined in sequential processes, we can assume there is only one process $p \in Pset$.) For each $RBj \in RBset$ (or $NVj \in NVset$) we define

- the final acceptance test $VTest_j$ (for RB_j only),
- the number of alternates n_j, and
- for each alternate, $CSec_{j,k}$ $(1 \le k \le n_j)$ (indicates the appropriate procedures).

CVset — Set of the conversations defined in the application program. For each $CVj \in CVset$ we define

- the set of participating processes $CVset_j = \{p_i\}$ (this set is the same for every alternate of CV_j),
- the set of those processes actually participating in CV_j at any given instant $CVentered_j$ (since they can enter asynchronously),
- the final acceptance test $VTest_{i,j}$ for each process,
- the number of alternates n_j, and
- $CSec_{i,j,k}$ $(1 \le k \le n_j)$, indicating one alternate procedure (*CodeSection* type).

*id*1, *id*2, etc. — Variables of the *Environment* type.

Figure 7. RMP data structures.

```
RBproc = ( id1 := Save(p); k := 1; bool := false;
           ((¬bool) & (k ≤ n)) * ( id2 := Execute(p, CScc_k, id1);
                                   bool := Evaluate (p, VTest, id2);
                                   k := k + 1);
           Discard(id1);
           if ¬bool then Terminate(p) else Continue(p, exit, id2))
```

Figure 8. A generalized process, RBproc, to execute any recovery block in any process.

egy (recovery block, *N*-version programming, conversation, or programmer transparent coordination), each monitoring its corresponding program portion.

Recovery block. Implementing a recovery block requires properly organizing the basic actions: saving a process' state, executing the nth alternate, executing the acceptance test, and discarding or restoring a saved state. We can use a generalized process, RBproc, to execute any recovery block in any process (see Figure 8).

When RBproc starts, it saves the current environment of process p and assigns its value to *id*1. The result of the execution of the first alternate defined in the recovery block is *id*2. The related acceptance test is then executed using *id*2. If the result is "true" (that is, if the alternate executes successfully), environment *id*1 is discarded and process p continues in environment *id*2. Otherwise, the next alternate executes, again in environment *id*1.

We implement the rollback and the resumption of the original checkpointed environment by making the environment in which a process must execute a parameter of the Execute function. Retries continue until an alternate succeeds or until no more alternates are available. In the latter case, the RMP could perform any suitable action; in Figure 8, the Terminate function is invoked for the sake of simplicity.

The kernel calls to Execute and Evaluate suspend RMP execution and transfer control to process p. Specifically, the Execute call makes the $CSec_k$ procedure of p execute in *id*1. Control then returns to the next RMP statement (the Evaluate call).

RBproc must be activated each time the application program enters the recovery block, that is, each time the application program is executing at some recovery block entry point. Assuming that only one recovery block is present in process p and that its entry point is labeled "entry," we can suspend the application program and transfer control to RBproc by including the following statement in the RMP main program:

at entry → RBproc

Now consider a process p in which several disjoint recovery blocks have been defined or in which a recovery block is defined inside an iteration. It is difficult or impossible to know the order in which these recovery blocks will be activated. We must generalize the simple structure of the RMP main program above. The choice statement makes this possible.

Let's call RB_j one of the recovery blocks defined in *RBset* (see Figure 7). Since one RMP process must be implemented for each RB_j in *RBset*, the RMP main program must contain the following choice statement:

$*(\ |_{RB_j \in RBset}$ at $entry_j \rightarrow RBproc_j)$

where each $RBproc_j$ is a jth instance of RBproc.

The choice construct lets us express RMP processes without knowing the application program's control flow. While the RMP main program is suspended on the above guarded statement, the application processes continue to work until one of the $entry_j$ addresses is attained, making the corresponding guard successful and forcing the execution of the RMP process $RBproc_j$. At the end of $RBproc_j$, the RMP process again waits for a possible subsequent execution of this or some other recovery block.

Figure 9 shows the control flow between the RMP and the application program for the recovery block in Figure 2, assuming that the first alternate fails. Note that Figure 9 is a refinement of Figure 1. Figure 9 also shows that, compared with other implementations of the same fault-tolerance actions, the RMP incurs an additional cost in the form of extra context switches. On the other hand, the RMP data is protected from application program interference.

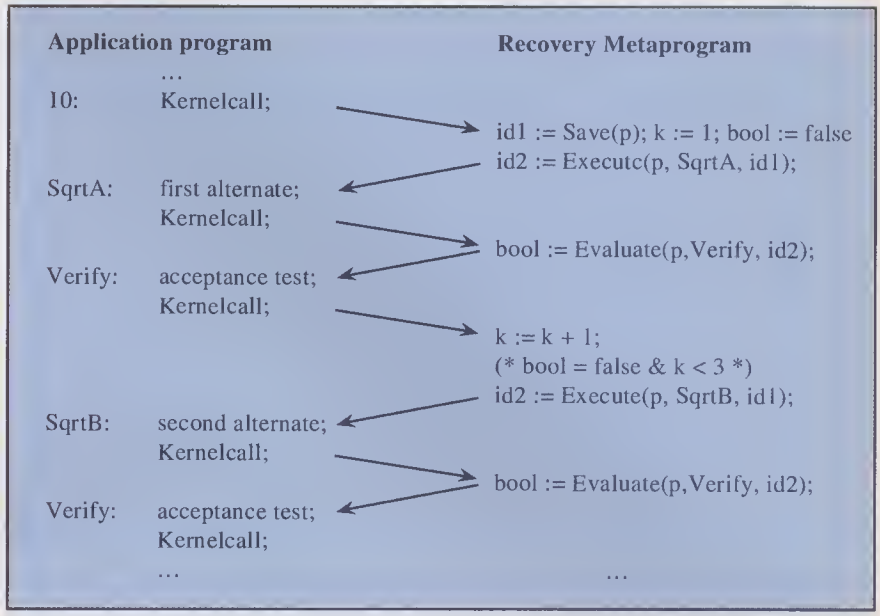


Figure 9. Control flow between the application program and RMP.

***N*-version programming.** We can implement *N*-version programming in a similar manner (see Figure 10). Note that the parallel construct on the Execute function implies that the same environment, *id*1, is replicated for each activated process. This implementation is not tied to a specific hardware architecture, that is, the parallel construct can be interpreted in different ways depending on the target. We must add at least one parameter to this kernel function to handle possible parallelism in the target.

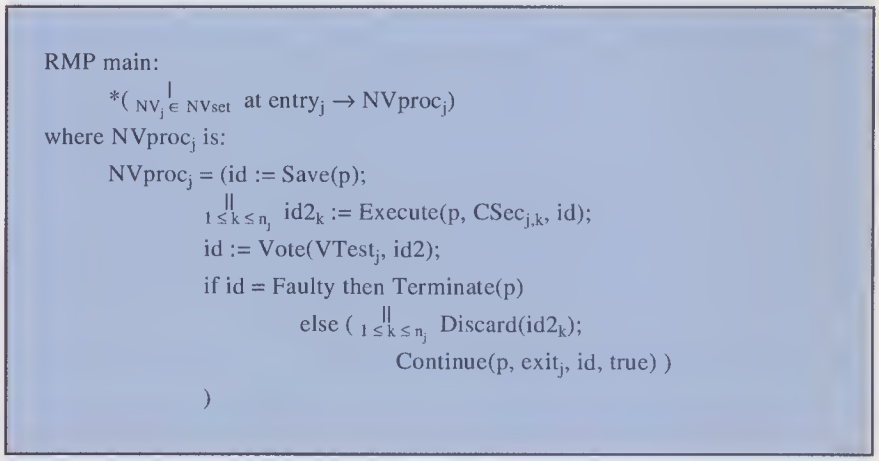


Figure 10. The *N*-version programming process.

Conversation. This implementation refers to the name-linked recovery blocks and assumes that there are no nested conversations, that participants for every alternate are the same and statically known, that participants enter the first alternate asynchronously and subsequent ones synchronously, and that the environment is centralized.

Figure 11 shows the process implementing conversation *CV*, which involves the set of processes *CVset*. If many conversations have been defined, each conversation CV_j will coordinate actions performed by processes in $CVset_j$.

We must implement the processes in Figure 11 differently for different implementations of the conversation scheme. For instance, nested conversations require manipulation of a stack of process sets, and a distributed implementation requires a two-phase "like" protocol to execute the global acceptance test.[6]

Programmer transparent coordination. Kim describes programmer transparent coordination[3] as a system where recovery blocks have been defined and processes interact through monitors, so that recovery actions involve monitor reference and update operations.

The algorithm initially considers only one monitor, p_m, which does not contain Wait or Signal instructions (see Figure 12). The monitor provides Update and Reference operations, with entry points named $p_m upd.entry$ and $p_m ref.entry$, respectively. Additional recovery points, called *branch RPs*, are established within each process and the monitor according to information received from other processes. At any given moment during program execution, we refer to the set of recovery blocks that has information on which process *X* depends as the direct potential recaller set, or *DPRS*(*X*). The algorithm also contains rules to discard or resume the branch RPs depending on the successful or failed execution of a recovery block.

Unlike the recovery block and conversation schemes, we cannot easily implement the PTC scheme in a structured way. The

```
RMP main:
*(CVentered := Ø;
  ( ||_{p_i ∈ CVset} at entry_i → CVproc_i);
  Conv)
where:
    CVproc_i = (id1_i := Save(p_i);
                CVentered := CVentered ∪ {p_i};
                (p_k ∈ CVentered) *( Restrict(p_k, CVentered, CVset));
                id2_i := Execute(p_i, CSec_{i,t}, id1_i))
    Conv = ( k := 1;  bool := false;
             (¬ bool) *( ||_{p_i ∈ CVset} b_i := Evaluate(p_i, VTest_i, id2_i))
                         bool := true;
                         (p_i ∈ CVset) *(bool := bool & b_i);
                         bool := bool or (¬ bool & (k = n));
                         k := k + 1;
                         if bool then skip
                                 else ||_{p_i ∈ CVset} id2_i := Execute(p_i, CSec_{i,k}, id1_i));
             if (k ≤ n) then Proceed else ||_{p_i ∈ CVset} Terminate(p_i)
           )

    Proceed = ( ||_{p_i ∈ CVset} (Discard(id1_i);
                                  Restrict(p_i, Pset, Pset);
                                  Continue(p_i, exit_i, id2_i, true))
              )

If more than one conversation is defined in the application program, we must
instantiate one such RMP process for each conversation:

*( ||_{CV_i ∈ CVset}   *(CVentered_j := Ø;
                         ||_{p_i ∈ CVset_j} at entry_{i,j} → CVproc_{i,j} ;
                         Conv_j )
 )
```

Figure 11. The conversation process.

CSP program in Figure 12 implements Kim's rules related to the set of branch RPs and the evaluation of the DPR set. The processes $Succ_{i,j}$ and $Fail_{i,j}$ are referred to but not shown in Figure 12; they perform the operations Kim defined.[3]

Since any of these four fault-tolerance strategies could be used in an application program, the RMP execution framework lets us apply the most suitable strategy for a particular situation. The RMP also lets us observe the application under different control policies to "fine tune" its behavior.

RMP modules implementing new fault-tolerance constructs and improved versions of old ones also have been developed, including a real-time extension of the recovery block scheme and a version of the conversation scheme that relaxes the synchronous acceptance test condition.

RMP implementation

Ozaki et al. have described a possible implementation of some of the primitives required by the RMP, using existing K286 primitives on 80286-based machines.[7] The K286 primitives include task management and scheduling operations, memory management support (segment creation and sharing), and intertask communication by use of mailbox primitives. Ozaki et al.showed how combining such primitives can support higher level operations used by the RMP.[7]

The programming environment. We have implemented a programming environment based on the Cross Multimicro Development System for multimicro embedded applications.[8] CMDS is based on the host-target approach and supports a high-level modular concurrent language, called the Multimicro Language (MML). (However, the actual organization of extensions to CMDS is largely independent from language or environment features.)

CMDS provides a set of tools on the host system based around a compiler and an allocator. The compiler can generate code for different target machines, while the allocator lets us assign the target system's physical resources to an MML program's logical entities. Runtime support for MML consists of a multiprocessor kernel devoted to basic I/O, message passing, and process scheduling.

We write fault-tolerant MML applications using Extended MML, a version of MML with a few syntactical constructs to define program parts that the RMP needs to supervise. Two tools have been added: a preprocessor for application programs and the RMP builder.

The preprocessor analyzes an Extended MML program and outputs an MML program plus information about its fault-tolerance needs. The compiler can then process the MML program, inserting kernel calls in the application program code corresponding with required switches between the application and the RMP (see Figure 9).

Using the fault-tolerance information from the preprocessor, the RMP builder extracts from the RMP library the appropriate modules to implement the selected fault-tolerance mechanisms and configures them into an RMP component. Figure 13 shows the information flow through such an environment.

Let's again consider the example in Figure 2. Assuming the two alternates (SqrtA and SqrtB) and the acceptance test (Verify) are written in MML, the preprocessor outputs a table naming these components as well as the entry and exit points. The RMP builder then inspects this table, extracts from the RMP library the recovery block module, and configures it as a process initialized with the collected information.

We can view the Recovery Metaprogram as a unifying mechanism that lets us implement different software fault-tolerance concepts in different contexts. As such, the RMP gives us a better understanding of how to incorporate fault-tolerance functions into application programs. We can use this insight, in turn, to develop improvements or extensions to existing mechanisms.

The reusability of the RMP processes is an additional advantage of the RMP approach, since most of the processes can be application independent. Separating the concerns of application design and development from those of fault-tolerance specification helps promote understanding of both and might result in increased system reliability. ■

Acknowledgments

This work was partially supported by the Italian National Research Council, under the grant, "Concurrent and Real-Time Programming," and is part of the joint research project "Fault-Tolerant Concurrent Software" between Florida Atlantic University and the Istituto per la Matematica Applicata.

References

1. B. Randell, "System Structure for Software Fault Tolerance," *IEEE Trans. Software Eng.*, Vol. SE-1, No. 2, June 1975, pp. 221-232.

2. L. Chen and A. Avizienis, "*N*-Version Programming: A Fault-Tolerant Approach to Reliability of Software Operation," *Proc. 8th Int'l Symp. Fault-Tolerant Computing*, 1978, Computer Society Press, Los Alamitos, Calif., Order No. 180 (microfiche only), pp. 21-23.

3. K.H. Kim, "Programmer Transparent Coordination of Recovering Concurrent Processes: Philosophy and Rules of Efficient Implementation," *IEEE Trans. Software Eng.*, Vol. SE-14, No. 6, June 1988, pp. 810-821.

4. K.H. Kim, "Approaches to Mechanization of the Conversation Scheme Based on Monitors," *IEEE Trans. Software Eng.*, Vol. SE-8, No. 3, May 1982, pp. 189-197.

5. C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, Englewood, N.J., 1985.

6. P. Jalote and R.H. Campbell, "Atomic Action for Fault Tolerance Using CSP," *IEEE Trans. Software Eng.*, Vol. SE-12, No. 1, Jan. 1986, pp. 59-68.

Data Structures:
$bRP_{i,j}$: A record, the first field of which represents an *Environment* and the second one an *Entrypoint*. It represents a branch RP inserted in process p_i and caused by RB_j, that is, if RB_j fails and rolls back, process p_i must roll back to the address and environment specified in $bRP_{i,j}$.

```
lab : Entrypoint variable;
PTCproc = ( . . .
  at RBentry_j →
      ( p_i := process involved in RB_j;
        DPRS(p_i) := DPRS(p_i) ∪ {RB_j};
        id_i,j := Save (p_i);
        Continue(p_i, entry_i,j, current));

  at p_mupd.entry →
      ( p_x := Caller(p_m); lab := Where_Is(p_m);
        DPset := DPRS(p_x) \ DPRS(P_m);
        if DPset = ∅ then skip
          else ( id := Save(p_m);
                 (RB_j ∈ DPset) * (bRP_m,j := (id,Nil));
                 DPRS(p_m) = DPRS(p_m) ∪ DPset;
                 Continue (p_m, lab, current)));

  at p_mref.entry →
      ( p_x := Caller(p_m); lab := Where_Is(p_m);
        DPset := DPRS(p_m) \ DPRS(p_x);
        if DPset = ∅ then skip
          else (id := Save(p_x);
                (RB_j ∈ DPset) * (p_k := process involved in RB_j;
                  bRP_k,x := (id,Where_Is(p_x)));
                DPRS(p_x) := DPRS(p_x) ∪ DPset;
                Continue (p_m, lab, current)));

  at RBexit_j →
      ( bool := Evaluate(p_i, VTest_i,j, current environment of p_i);
        if bool then Succ_i,j else Fail_i,j);
. . . )
```

Figure 12. Part of the programmer transparent coordination (PTC) process.

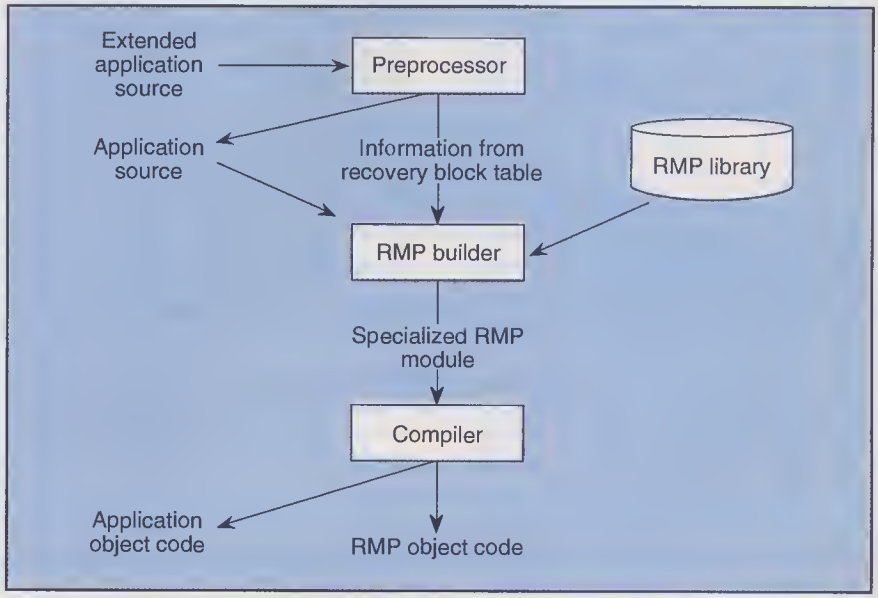


Figure 13. The programming environment.

7. B.M. Ozaki, E.B. Fernandez, and E. Gudes, "Software Fault Tolerance in Architectures with Hierarchical Protection Levels," *IEEE Micro*, Vol. 8, No. 4, Aug. 1988, pp. 30-43.

8. A. Clematis, G. Dodero, and V. Gianuzzi, "A Design Tool for Fault-Tolerant Software," *Proc. CompEuro 90*, Computer Society Press, Los Alamitos, Calif., Order No. 2041, pp. 130-137.

Gabriella Dodero is an assistant professor in the Department of Mathematics at the University of Genova, Italy. Her research interests include programming languages and environments, and distributed and fault-tolerant software.

Dodero received an advanced degree in mathematics from the University of Genova in 1977. She is a member of the IEEE Computer Society, the IEEE, and the Working Group on Dependable Computers of the Italian Association of Automatic Calculating.

Vittoria Gianuzzi is an assistant professor in the Department of Mathematics at the University of Genova, Italy, and a research consultant at the Institute of Applied Mathematics of the Italian National Research Council, Genova. Her research interests include programming languages, software fault tolerance, and compiler construction for parallel systems.

Gianuzzi received an advanced degree in mathematics from the University of Genova in 1975. She is a member of the Working Group on Dependable Computers of the Italian Association of Automatic Calculating.

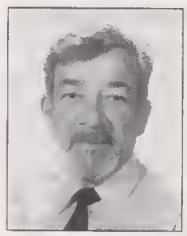
Massimo Ancona is an associate professor in the Department of Mathematics at the University of Genova, Italy. His research interests include design and implementation of programming languages, object-oriented and concurrent programming, compiler construction, software engineering and software fault tolerance, data structures, and algorithms.

Ancona received the doctoral degree in mathematics from the University of Genova in 1965. He is a member of the IEEE Computer Society and the ACM.

Readers can contact Ancona, Dodero, and Gianuzzi at Dipartimento di Matematica dell'Universita,' Via LB Alberti, 4, 16132 Genova, Italy. Clematis can be reached at Istituto per la Matematica Applicata del CNR, Via L.B. Alberti, 4, 16132 Genova, Italy. Fernandez is at the Department of Computer Engineering, Florida Atlantic University, Boca Raton, FL 33431.

Andrea Clematis is a senior scientist at the Institute of Applied Mathematics of the Italian National Research Council, Genova, Italy. He is also an adjunct professor in the Department of Computer Science at the University of Genova. His research interests include fault-tolerant software, programming languages and environments, and multiprocessor and distributed systems.

Clematis received an advanced degree in mathematics from the University of Genova in 1982. He is a member of the IEEE Computer Society, the ACM, and the Working Group on Dependable Computers of the Italian Association of Automatic Calculating.



Eduardo B. Fernandez is a professor in the Department of Computer Engineering at Florida Atlantic University, Boca Raton, Florida. His research interests include fault-tolerant systems, computer security, database systems, and computer architecture.

Fernandez holds an MS degree in electrical engineering from Purdue University and a PhD in computer science from UCLA. He has written three books and a number of papers. He is a member of the IEEE Computer Society.

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

COMPUTER ENGINEERING DEPARTMENT

The Computer Engineering Department seeks applications for faculty positions at all levels. Preference will be given to experienced applicants at the associate and full professorial ranks. Applicants must hold a Ph.D. degree in Computer Engineering or related areas. Individuals with demonstrated research records and teaching experience in one or more of the following areas will be considered: Fault Tolerant Computing, Data Communication and Computer Networks, VLSI and Design Automation, Robotics, Computer Architecture. Teaching and research at the Department are supported by a VAX 11/7800, a fully equipped, Computer Graphics Center, as well as a University Data Processing Center that has AMDAHL 5850 and IBM 3090 mainframes. In addition, research and teaching laboratories in the department includes: Design Automation Lab, Digital System Design Lab, Microprocessor Systems Lab, Printed Circuit Board Facility, Robotics Lab, and Computer Communication Networks Lab.

KFUPM offers attractive salaries commensurate with qualifications and experience, and benefits that include free furnished airconditioned accommodation on campus, yearly repatriation tickets, ten months duty each year with two months vacation salary. Minimum regular contract for two years, renewable.

Interested applicants are requested to send their Curriculum Vitae with supporting information not later than one month from the date of this publication, to: **DEAN OF FACULTY AND PERSONNEL AFFAIRS**
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DEPT NO. 9052
DHAHRAN 31261, SAUDI ARABIA

Fault-Tolerant Clock Synchronization in Distributed Systems

Parameswaran Ramanathan, University of Wisconsin

Kang G. Shin, University of Michigan

Ricky W. Butler, NASA Langley Research Center

Digital computers have become essential to critical real-time applications such as aerospace systems, life support systems, nuclear power plants, drive-by-wire systems, and computer-integrated manufacturing systems. Common to all these applications is the demand for maximum reliability and high performance from computer controllers. This requirement is necessarily stringent because a single controller failure in these applications can lead to disaster. For example, the allowable probability of failure for a commercial aircraft is specified to be less than 10^{-9} per 10-hour mission because a controller failure during flight could result in a crash.

Because of such stringent requirements, traditional methods for design and validation of computer controllers are often inadequate. Ad hoc techniques that appear sound under a careful failure-modes-and-effects analysis are often susceptible to certain subtle failure modes. The clock synchronization problem shown in Figure 1 is a classic example. The figure shows a three-node system in which each node has its own clock. The clocks are synchronized by adjusting each to the median of the three clock values. This "intuitively correct" algorithm works fine as long as all the clocks are consistent in their behavior, as Figure 1a illustrates. However, if one clock is faulty and misinforms the other two clocks, the two nonfaulty clocks cannot be synchronized. For example, in Figure 1b the faulty clock *B* reports incorrectly to clocks *A* and *C*. As a result, clocks *A* and *C* do not make any corrections because both behave as if they are the median clock.

Software algorithms are suitable only where loose synchronization is acceptable, and hardware algorithms are expensive. A hybrid scheme achieves reasonably tight synchronization and is cost-effective.

Lamport and Melliar-Smith were the first to study the three-clock synchronization problem in the presence of arbitrary fault behavior.[1] They coined the term Byzantine fault to refer to the fault model in which a faulty clock can exhibit arbitrary behavior including, but not limited to, misrepresenting its value to other clocks in the system. They showed that in the presence of Byzantine faults, no algorithm can guarantee synchronization of the nonfaulty clocks in a three-node system. They also showed that $3m + 1$ clocks are sufficient to ensure synchronization of the nonfaulty clocks in the presence of m Byzantine faults. This condition later proved not only sufficient but also necessary for ensuring synchronization in the presence of Byzantine faults.

Since the initial study by Lamport and Melliar-Smith, the problem of clock synchronization in the presence of Byzantine faults has been studied extensively by several other researchers. All this attention is mainly be-

0018-9162/90/1000-0033$01.00 © 1990 IEEE

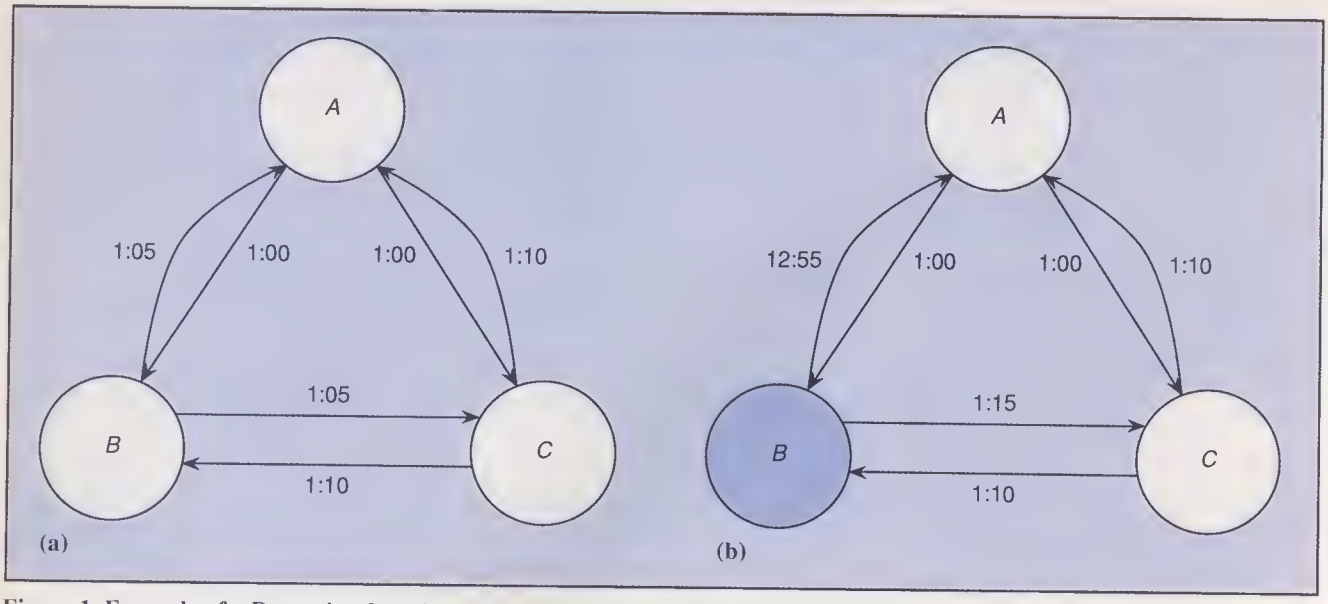


Figure 1. Example of a Byzantine fault in clock synchronization: (a) all clocks nonfaulty; (b) faulty clock at node *B*.

cause many applications require a consistent view of time across all nodes of a distributed system, and this can be achieved only through clock synchronization.

Solutions proposed in the literature on clock synchronization take either a software or a hardware approach. The software approach is flexible and economical, but additional messages must be exchanged solely for synchronization.[1-4] Because they depend on message exchanges, the worst-case skews guaranteed by most of these solutions are greater than the difference between the maximum and minimum message transit delay between any two nodes in the system. The hardware approach, on the other hand, uses special hardware at each node to achieve a tight synchronization with minimal time overhead.[5-8] However, the cost of additional hardware precludes this approach in large distributed systems unless a very tight synchronization is essential. Hardware solutions also require a separate network of clocks that is different from the interconnection network between the nodes of the distributed system.[6]

Because of these limitations in the software and hardware approaches, researchers have begun investigating a hybrid approach. A hardware-assisted software synchronization scheme has been proposed that strikes a balance between the hardware requirement at each node and the clock skews attainable.[9] Another recently proposed approach is probabilistic clock synchronization, in which the worst-case skews can be made as small as desired.[10] However, depending on the desired worst-case skews, there is a nonzero probability of loss of synchronization. Also, this approach may induce very high traffic to the system.

This article compares and contrasts existing fault-tolerant clock synchronization algorithms. The worst-case clock skews guaranteed by representative algorithms are compared, along with other important aspects such as time, message, and cost overhead imposed by the algorithms. Special emphasis is given to more recent developments such as hardware-assisted software synchronization, probabilistic clock synchronization, and algorithms for synchronizing large, partially connected distributed systems.

Preliminary concepts

First we define some of the concepts common to most clock synchronization algorithms and introduce the notation that will be used throughout the article (see sidebar on next page). We begin with the notion of a clock.

> Definition 1: Time that is directly observable in some particular clock is called its clock time. This contrasts with the term real time, which is measured in an assumed Newtonian time frame that is not directly observable.

It is convenient to define the local clock at a node as a mapping from real time to clock time. In other words, let C be a mapping from real time to clock time, then $C(t) = T$ means that when the real time is t, the clock time at a particular node is T. We adopt the convention of using lowercase letters to denote quantities that represent real time and uppercase letters to denote quantities that represent clock time. Figure 2 illustrates the concept of a clock function/mapping. A perfect clock is one in which a unit of clock time elapses for every unit of real time. If more or less than one unit of clock time elapses for every unit of real time, the clock is said to be fast or slow, respectively.

Since a properly functioning clock is a monotonic increasing function, its inverse function is well defined. Let $c(T) = C^{-1}(T) = t$ denote this inverse function. In the literature some of the results in clock synchronization are formulated using the clock function, while others are formulated using the inverse function. We will use subscripts to distinguish between the different clocks in the system. For example, $C_p(t)$ will denote the clock at node p, and $C_q(t)$ the clock at node q. A clock is considered nonfaulty if there is a bound on the amount of deviation from real time for any given finite time interval. In other words, even nonfaulty clocks do not always maintain perfect time.

> Definition 2: A clock c is said to be nonfaulty during the real-time interval $[t_1, t_2]$ if it is a monotonic, differentiable

function on $[T_1, T_2]$ where $c(T_i) = t_i$, $i = 1,2$, and for all $T \in [T_1, T_2]$

$$\left| \frac{dc(T)}{dT} - 1 \right| < \frac{\rho}{2}$$

for some constant ρ. The constant ρ is said to be the *drift rate* of the nonfaulty clock.

Instead of the above definition, some synchronization algorithms are defined using one of the following near-equivalent definitions for a nonfaulty clock:

Definition 2a: A clock c is said to be nonfaulty if there exists a constant ρ_2 such that for t_1 and t_2

$$1 - \rho_2 < \frac{C(t_2) - C(t_1)}{t_2 - t_1} < 1 + \rho_2$$

Definition 2b: A clock c is said to be nonfaulty if there exists a constant ρ_3 such that for t_1 and t_2

$$\frac{1}{1 + \rho_3} < \frac{C(t_2) - C(t_1)}{t_2 - t_1} < 1 + \rho_3$$

The near equivalence of these definitions follows from the Taylor series expansion of $(1 + \rho)^{-1}$:

$$(1 + \rho)^{-1} = 1 - \rho + \rho^2 - \rho^3 + \rho^4 - \ldots$$

For a typical value of $\rho = 10^{-6}$, the second-order and high-order terms can be ignored, thus implying $\rho_2 = \rho_3 = \rho/2$.

The above notion of a clock does not imply any particular implementation. In fact, some synchronization algorithms deal with hardware clocks, while others deal with logical clocks. Hardware clocks are the actual clock pulses that control circuitry timing; logical clocks are values derived from hardware clocks. For instance, a logical clock might be the value of a counter that is incremented once every predetermined number of pulses of the hardware clock. In either case, we can talk about synchronization in terms of the abstract notion of the mapping function from clock time to real time. The main difference will be in the granularity, or skew, of synchronization.

Definition 3: Two clocks c_1 and c_2 are said to be *δ-synchronized* at a clock time T if and only if $|c_1(T) - c_2(T)| \leq \delta$. A set of clocks are said to be *well synchronized* if and only if any two nonfaulty clocks in this set are δ-synchronized for some specified constant δ.

Notations used in this article

$c_p(t)$	Clock time at node p when the real time is t
$C_p(T)$	Real time when the clock time at node p is T
ρ	Maximum drift rate of all clocks in the system
δ	Maximum skew between any two nonfaulty clocks in the system
ε	Upper bound on the read error
m	Maximum number of faulty clocks in the system
N	Total number of clocks in the system
R	Resynchronization interval
U	Upper bound on message transit delay
Δ_{qp}	Node p's perception of its skew with respect to clock at node q

Because of the nonzero drift rates of all clocks, a set of clocks does not remain well synchronized without some periodic resynchronization. This means that the nodes of a distributed system must periodically resynchronize their local clocks to maintain a global time base across the entire system. Synchronization requires each node to read the other nodes' clock values. The actual mechanism used by a node to read other clocks differs from one algorithm to another. In hardware algorithms the clock signal from each of the other nodes (or an appropriate subset of nodes) is an input to the synchronization circuitry at each node. In software algorithms each node either broadcasts its clock value to all nodes at specified times or sends its clock value individually to requesting nodes.

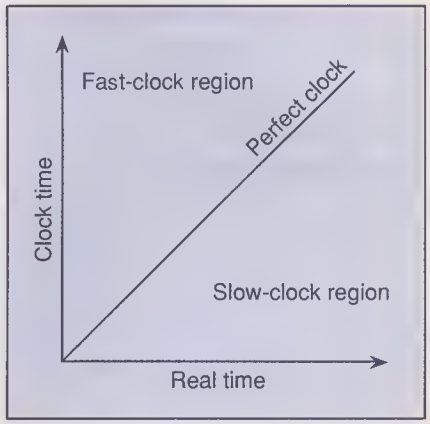


Figure 2. The clock function.

Regardless of the actual reading mechanism, a node can obtain only an approximate view of its skew with respect to other nodes in the system. Errors occur mainly because it takes a finite and unpredictable amount of time to deliver a clock signal or a clock message from one node to another. In hardware algorithms, errors are due mainly to the unpredictable propagation delays for clock signals, whereas in software algorithms errors are due to variation in the message transit delays. Most of the synchronization algorithms discussed in this article are based on the assumption that if two nodes are nonfaulty, the error in reading each other's clock is bounded. Since the actual errors that occur differ from one algorithm to another, we will discuss them further as we describe each algorithm.

The time at which a node decides to read the clocks at other nodes also depends on the algorithm under consideration. In hardware algorithms, synchronization circuitry continuously monitors the frequency and phase of all clocks. On the basis of the input signals, the circuitry also updates the local clock continuously. Hardware algorithms can therefore be classified as continuous-update algorithms. Software synchronization algorithms, on the other

hand, are usually discrete-update algorithms; that is, correction to a local clock is computed at discrete time intervals. However, software algorithms may differ in the way they apply this correction to the local clock. In some software synchronization algorithms the correction is applied instantaneously, whereas in others it is spread over a time interval.

The time at which the correction is computed is determined by each node on the basis of its own clock. The time interval between successive corrections is called the resynchronization interval, denoted R, and is usually a constant known to all nodes. If $T^{(0)}$ denotes the time at which the system began its operation, then the time of the ith resynchronization is $T^{(i)} = T^{(0)} + iR$. The time interval $R^{(i)} \equiv [T^{(i)}, T^{(i+1)}]$ is often called the ith resynchronization interval. In discrete-update algorithms, we can view the local clock of a node as a series of functions, one for each resynchronization interval. That is, the clock at node p in the $(i + 1)$th resynchronization is given by

$$c_p^{(i+1)}(T) = c_p^{(0)}(T + \xi_p^{(i)})$$

where $\xi_p^{(i)}$ represents the change in p's clock since the start of the system.

Software synchronization

The basic idea of software synchronization algorithms is that each node has a logical clock that provides a time base for all activities on that node. This logical clock is derived from the hardware clock on that node, though it usually has a much larger granularity than the hardware clock. The algorithm executed by each node for synchronizing the logical clocks can be viewed as a clock process invoked at the end of every resynchronization interval. This clock process is responsible for periodically reading the clock values at other nodes and then adjusting the corresponding local clock value.

Informally, any software synchronization algorithm must satisfy the following two conditions:

Agreement: The skew between all nonfaulty clocks in the system is bounded.

Accuracy: The logical clocks keep up with real time.

The first condition states that there is a consistent view of time across all nodes in the system. The second condition states that this view of time is consistent with what is happening in the environment. The synchronization algorithms differ in the way these two conditions are specified, as well as in the way the clock processes read the clock values at other nodes. In some algorithms, a clock process requests and receives a clock value from each of the other nodes, while in other algorithms the clock process broadcasts its local clock value when it thinks it is time for a resynchronization. The other nodes receive the broadcast message and use the clock value for correcting themselves at a later time. In either case, the skew perceived by a receiving clock process differs from the actual skew that exists between the two clocks, because of the errors in reading the clocks. These errors are due mainly to unpredictable variation in the communication delay between the two nodes. This notion is formalized in the following discussion.

Let p and q be any two nonfaulty nodes in the system. Let T be the clock time of node q when the clock process at node q initiates a broadcast of the local clock value. In other words, let the clock process at node q initiate a broadcast of the local clock value at real time $c_q(T)$. This message will reach p after some time delay, say Q. That is, the broadcast message is delivered to node p at real time $c_q(T) + Q$. At that instant, the clock times at nodes p and q are $C_p(c_q(T) + Q)$ and $C_q(c_q(T) + Q)$, respectively. That is, the actual skew existing between p and q at the instant when p receives this message is given by $C_q(c_q(T) + Q) - C_p(c_q(T) + Q)$. However, node p has no way of determining $C_q(c_q(T) + Q)$ exactly. The best it can do is estimate the delay Q by, say, $1 + \rho_3\hat{Q}$, and compute the skew as follows:

$$\Delta_{qp} = T + \hat{Q} - C_p(c_q(T) + Q).$$

This means that the error in the perceived skew at p is

$$e_{qp} = T + \hat{Q} - C_q(c_q(T) + Q).$$

This error is small if we can obtain a good estimate of the communication delay and if the clock drifts of p and q are comparable, especially during the communication delay. All synchronization algorithms discussed in this article are based on the assumption that if p and q are nonfaulty, then e_{qp} is bounded.

From the above discussion we can conclude that the read error is not a serious problem if the algorithm is used to synchronize a fully connected system. However, it can be a serious limitation in future distributed systems because system size is increasing, and it is difficult to fully connect all the nodes in a large distributed system. This problem has been specifically addressed by some of the recent algorithms.[9,10]

Software synchronization algorithms can be classified as convergence algorithms with averaging, convergence algorithms without averaging, or consistency algorithms (see Figure 3). The leaves of the tree in Figure 3 are the representative software synchronization algorithms surveyed in this article. (For additional reading refer to Schneider.[11])

Convergence-averaging algorithms. Convergence-averaging algorithms are based on the following idea: The clock process at each node broadcasts a "resync" message when the local time equals $T^{(0)} + iR - S$ for some integer i and a parameter S to be determined by the algorithm. After broadcasting the clock value, the clock process waits for S time units. During this waiting period, the clock process collects the resync messages broadcast by other nodes. For each resync message, the clock process records the time, according to its clock, when the message was received. At the end of the waiting period, the clock process estimates the skew of its clock with respect to each of the other nodes on the basis of the times at which it received resync messages. It then computes a fault-tolerant average of the estimated skews and uses it to correct the local clock before the start of the next resynchronization interval.

The convergence-averaging algorithms proposed in the literature differ mainly in the fault-tolerant averaging function used to compute the correction. In algorithm CNV,[1] each node uses the arithmetic mean of the estimated skews as its correction. However, to limit the impact of faulty clocks on the mean, the estimated skew with respect to each node is compared against a threshold, and skews greater than the threshold are set to zero before computing the mean. In contrast, with the algorithm in Lundelius-Welch and Lynch[3] (henceforth referred to as algorithm LL), each node limits the impact of faulty clocks by first discarding the m highest and m lowest estimated skews and then using the midpoint of the remaining skews as its correction, where m is the maximum number of faulty clocks to be tolerated.

One main limitation of the convergence-averaging algorithms is that they are in-

tended for a fully connected network of nodes. As a result, the algorithms are not easily scalable. In addition, they all require known upper bounds on clock read error and initial synchronization of the clocks. The need for initial synchronization is not a serious problem, because there are several algorithms to ensure it.[3] However, the assumption about the bound on the clock read error is a serious problem because the worst-case skews guaranteed by these convergence-averaging algorithms are usually greater than the bound on the read error.

In addition to the read error, the worst-case skews depend on the time interval between the resynchronizations, the clock drift rates, the number of faults to be tolerated, the total number of clocks in the system, and the duration of the time interval during which the clock processes read the clock values at other nodes. However, among all these factors, the read error has the greatest impact on the worst-case skew. Lamport and Melliar-Smith[1] show the worst-case skew for algorithm CNV to be

$$\max\{[N/N-3m][2\varepsilon+\rho(R+2[(N-m)/N]S)],\ \delta_0+\rho R\}$$

where δ_0 is the maximum skew after the initial synchronization, N is the total number of clocks in the system, m is the maximum number of faults to be tolerated, S is the duration of the time interval during which clock processes read the clock values at other nodes, and ε is the assumed bound on the read error. Similarly, in Lundelius-Welch and Lynch[3] the worst-case skew of algorithm LL was shown to be

$$\beta+\varepsilon+\rho(7\beta+U+4\varepsilon)+4\rho^2(2+\rho)(\beta+U)$$

where U is the maximum message transit delay between any two nodes in the system, ε is such that $U-2\varepsilon$ is the minimum message transit delay between any two nodes in the system, and β is roughly $4(\varepsilon+\rho R)$.

Since, typically, ρ is on the order of 10^{-6}, R is on the order of a few seconds, U and S are on the order of a few milliseconds, and ε is on the order of a few microseconds, $\rho U \ll \varepsilon$ and $\rho\varepsilon \ll \varepsilon$. Dropping the higher order terms, the worst-case skew of algorithm CNV[1] is approximately

$$[N/(N-3m)](2\varepsilon+\rho R)$$

while the worst-case skew of the algorithm LL[3] is $5\varepsilon+4\rho R$. Superficially, this seems to indicate that algorithm LL is better than algorithm CNV. However, because of the nature of the clock-reading process, the read error in algorithm LL is much larger than that in algorithm CNV, and the worst-case skews of the two algorithms are actually comparable. This is an example of the many pitfalls encountered in comparing clock synchronization algorithms.

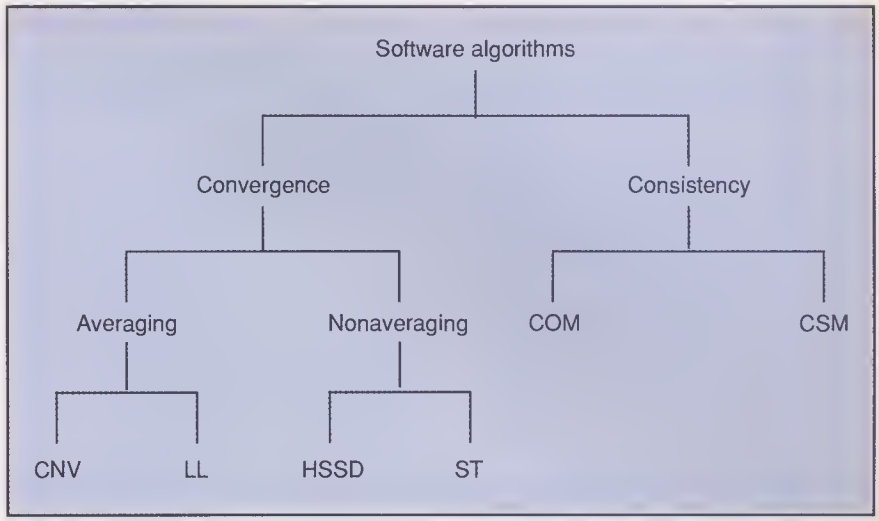


Figure 3. Classification of software synchronization algorithms.

Convergence-nonaveraging algorithms. Like algorithms CNV and LL, convergence-nonaveraging algorithms are discrete-update algorithms. However, they do not use the principle of averaging to synchronize nonfaulty clocks. Instead, each node periodically seeks to be the system synchronizer. All the nonfaulty nodes know the time at which the nodes try to become the system synchronizer. If all the nodes are nonfaulty, only one becomes the synchronizer. If the synchronizer fails, the algorithm is designed so that the remaining nonfaulty nodes effectively take over and synchronize despite the faulty synchronizer's erroneous behavior.

Like convergence-averaging algorithms, convergence-nonaveraging algorithms also require initial synchronization and a bound on the maximum message transit delay in the system. However, convergence-nonaveraging algorithms do not require a fully connected network of nodes. Instead, they require an authentication mechanism that the nodes can use to encode their messages in such a way that no other node can generate the same message or alter the message without detection. This can be achieved by using either digital signatures[2] or an appropriate broadcast algorithm.[4] As long as all nonfaulty nodes can communicate with each other, the convergence-nonaveraging algorithms will ensure that they are synchronized. However, the worst-case skew between the nonfaulty clocks is a strong function of the maximum message transit delay in the system. This means that as the connectivity decreases, the worst-case skew guaranteed by the algorithms increases.

In convergence-nonaveraging algorithms, a node resynchronizes its clock either when its local clock reaches the time for the next resynchronization or when it receives a signed message from other nodes indicating that they have resynchronized their clocks. To prevent faulty nodes from falsely triggering a resynchronization, each node performs a validity check before reacting to any message. The validity check's exact nature depends on the algorithm. In Halpern et al.,[2] a node considers a resynchronization message to be valid and hence is willing to resynchronize even before its clock reaches the time for the next resynchronization only if

(1) all signatures in the message are valid, indicating that the message is authentic;
(2) the time stamp on the message corresponds to the time for the next resynchronization — that is, for the ith resynchronization, the message should have a time stamp $T^{(i)} = T^{(0)} + iR$; and
(3) the message is received sufficiently close to the time for resynchronization — that is, a message with k distinct signatures should be received when the local time lies in the inter-

val $(T^{(i)} - kU, T^{(i)})$, where U is the maximum message transit delay between any pair of nodes in the system.

In Srikanth and Toueg,[4] however, a node considers all resynchronization messages suspicious. Therefore, a node is willing to resynchronize before its clock reaches the time for the next resynchronization only if it receives a message from $m + 1$ other nodes indicating that they have resynchronized. This is done to ensure that at least one nonfaulty node's clock has reached the time for resynchronization.

The main limitation of convergence-nonaveraging algorithms is that the worst-case skew is greater than the maximum message transit delay between any pair of nodes. For instance, the worst-case skews of the algorithms in Halpern et al.[2] and Srikanth and Toueg[4] are $(1 + \rho)U + \rho(2 + \rho)R$ and $[R(1 + \rho) + U][\rho(2 + \rho)/(1 + \rho)] + U(1 + \rho)$, respectively. Since all the terms are positive, the worst-case skew is greater than the maximum message transit delay, U, in both algorithms. Since U can be on the order of tens of milliseconds, especially in a large, partially connected distributed system, the worst-case skews offered by these algorithms are also on the order of tens of milliseconds, which may not be acceptable in many applications.

One of the unique features in Srikanth and Toueg's algorithm is that the accuracy of the logical clocks is guaranteed to be optimal in the sense that the drift rate of the logical clocks is the same as the drift rate of the underlying hardware clocks. This is not necessarily the case with other software clock synchronization algorithms, where the logical clocks are guaranteed only to be within a linear envelope of the hardware clocks. Furthermore, in contrast to the algorithm in Halpern et al., Srikanth and Toueg's algorithm does not require a clock value to be sent in a resynchronization message. This characteristic can be used to eliminate some of the possible failure modes observed in a clock synchronization scheme.

Consistency algorithms. Consistency-based algorithms use a completely different principle than convergence-based algorithms. They treat clock values as data and try to ensure agreement by using an interactive consistency algorithm.[12] This is a distributed algorithm that ensures agreement among nonfaulty nodes on the private value of a designated sender node through a series of message exchanges. By "agreement" we mean the following two conditions are satisfied:

(1) All nonfaulty nodes agree on the sender's private value.

(2) If the sender is nonfaulty, the value agreed on by the nonfaulty nodes equals the sender's private value.

Note that nonfaulty nodes must agree on the sender's private value regardless of whether the sender is faulty or not. However, if the sender is faulty, the value agreed on by the nonfaulty nodes need not equal the sender's private value.

With consistency-based algorithms, at the end of every resynchronization interval each node treats its local time as a private value and uses an interactive consistency algorithm to convey this value to other nodes. From the clock values thus obtained from all the other nodes, each node computes an estimate of the skew with respect to every other node. Each node then uses the median skew to correct the local clock at the start of the next resynchronization interval.

Like convergence algorithms, consistency-based algorithms require a bound on the read error. Read errors occur in these algorithms because there can be a slight difference between the times at which two nodes, say p and q, decide on the clock value of another node, say r. Consequently, although p and q will agree on the clock value of r, the estimate of the skew they compute between their own clocks and r's clock might be slightly different. In addition to a bound on the read error, consistency-based algorithms also require certain conditions for correctly executing an interactive consistency algorithm. These requirements include conditions on the minimum number of nodes, sufficient connectivity, and a bound on the message transit delay. Lamport, Shostak, and Pease provide more details on these requirements.[12]

Note that these algorithms require no assumption about initial synchronization. Nor do they require a direct connection between all the nodes, although the algorithms described by Lamport and Melliar-Smith[1] do require such a connection.

The following example should clarify the basic idea of these algorithms. Consider a four-node system with a maximum of one Byzantine fault. In an interactive consistency algorithm,[12] a node p would convey its private value to other nodes using the following scheme: Node p would send its value to every other node, which in turn would relay the value to the two remaining nodes. That is, each node would receive three "copies" of p's value: one directly from p and the other two from the other two nodes. Each node would then estimate p's value to be the median of the values in the three copies. This scheme can be modified for clock synchronization by letting each node independently use the interactive consistency algorithm to convey its clock value to other nodes. At the end of four applications of the interactive consistency algorithm (one for each node), the local clock at each node could be adjusted to equal the median of the other nodes' clock values.

This clock synchronization scheme can be further extended to situations with more than one Byzantine fault, just like the interactive consistency algorithm.[12] However, this will require at least $m + 1$ rounds of message exchange, where m is the maximum number of faults to be tolerated, and this implies more overhead. The worst-case skew guaranteed by consistency-based algorithms, however, is usually smaller than those guaranteed by the convergence-based algorithms. For example, if the total number of clocks in the system equals $3m + 1$, then the worst-case skew of algorithm COM in Lamport and Melliar-Smith was shown to be $(6m + 4)\varepsilon + (4m + 3)\rho S + \rho R$ as opposed to $(6m + 2)\varepsilon + (4m + 2)\rho S + (3m + 1)\rho R$ for algorithm CNV,[1] where ε, S, and R are as described earlier under "Convergence-averaging algorithms."

The number of messages required for consistency-based algorithms can be reduced by using digital signatures to authenticate the messages. This is true of algorithm CSM in Lamport and Melliar-Smith. Algorithms with authentication require fewer messages and fewer nodes to tolerate a given number of faults. They also achieve a tighter skew than algorithms that do not use authentication.

Probabilistic synchronization. The main limitation of the algorithms discussed above is that the worst-case skew depends heavily on the maximum read error. This can be a serious problem in future distributed systems because read errors are likely to increase with system size. To remedy this problem, Cristian proposed a probabilistic synchronization scheme in which the worst-case skews can be made as small as desired.[10] However, the overhead imposed by the synchronization algorithm increases drastically as we reduce the skew. Furthermore, unlike other algorithms we have discussed, this algorithm does not guarantee synchronization with probability one. In fact, there is a nonzero probabil-

ity of loss of synchronization, and this probability increases with a decrease in the desired skew.

Cristian's idea is to assume that the probability distribution of message transit delay is known and let each node make several attempts to read the other clocks. After each attempt, the node can calculate the maximum error that might occur if the clock value obtained in that attempt were used to determine the correction. By retrying often enough, a node can read the other clocks to any given precision with probability as close to one as desired. This scheme is particularly suitable for systems having a master-slave arrangement in which one clock has been designated or elected master and the other clocks act as slaves.

More specifically, each node periodically sends a message to the master node requesting its clock value. When the master receives this request, it responds with a message saying "The time is T." When the response reaches the requesting node, it calculates the total round trip delay according to its own clock. If D is the round trip delay as measured by a node p, and if q is the master node, then Cristian proved that if nodes p and q are nonfaulty, the local time at node q when p receives the response message lies in the interval

$$[T + U_{min}(1 - \rho),\ T + D(1 + 2\rho) - U_{min}(1 + \rho)]$$

where U_{min} is the minimum message transit delay between p and q. Therefore, the maximum read error is $D(1 + 2\rho) - 2U_{min}$.

It follows from this result that if the round trip delay is small (close to $2U_{min}$), then so is the read error. Cristian's algorithm uses this observation to limit the read error. Each node is allowed to read the master's clock repeatedly until the round trip delay is such that the maximum read error is below a given threshold. Once a node reads the master's clock to the desired precision, it sets its clock to that value.

One limitation of this approach is the need to restrict the number of read attempts, since these are directly related to the overhead imposed by the algorithm. This implies that a node may not always be able to read the master's clock to the desired precision and hence could result in loss of synchronization. A second limitation is that it is not fault tolerant, since it is not easy to detect the master's failure. Furthermore, algorithms to designate or elect a new master are fairly complex and time-consuming.

Hardware synchronization

The principle of hardware synchronization algorithms is that of a phase-locked loop. The hardware clock at each node is an output of the voltage-controlled oscillator. The voltage applied to the oscillator comes from a phase detector whose output is proportional to the phase error between the phase of its clock (the output of the voltage-controlled oscillator it is controlling) and a reference signal generated by using the other clocks in the system. Thus, by adjusting the frequency of each individual clock to the reference signal, the clocks can always be kept in lock-step with respect to one another.

Algorithms. One of the first hardware synchronization algorithms resilient to Byzantine faults was proposed originally by T.B. Smith. This algorithm was developed to synchronize the processors of the fault-tolerant multiprocessor (FTMP).[13] The FTMP uses only four clocks and was expected to tolerate a maximum of one Byzantine fault, so Smith's algorithm was specifically developed for that situation. In the algorithm, each clock observes the other three clocks continuously and determines its phase difference with each of them. These phase differences are arranged in an ascending order, and the clock corresponding to the median phase difference is selected as the reference signal. Each clock uses a phase-locked loop to synchronize itself with the reference signal it selected.

The worst-case skew in this algorithm depends on the characteristics of the phase-locked loop. Smith did not characterize this worst-case skew, but experimental results on the FTMP have shown that the skews are around 50 nanoseconds. This should be compared with the skews in the software synchronization algorithms, which are on the order of microseconds. Surprisingly, this algorithm does not easily generalize to a situation in which more than one Byzantine fault must be tolerated. This is because with a median select algorithm it is possible for Byzantine failures to partition the set of clocks into two or more separate "cliques" that are internally synchronized but are not synchronized with other cliques.

As an illustration, consider a system of seven clocks. Such a system should be able to mask the failure of two clocks. Suppose that the five nonfaulty nodes (clocks) are named a, b, c, d, and e and the faulty nodes (clocks) are named x and y. If the transmission delays between clocks are negligible, then the order of arrival of the clock signals from the nonfaulty nodes should be the same at all the nodes. If the faulty clocks behave erratically, their order may be seen differently by different nodes. Consider the following ordering of signals as seen by the various nodes in the system:

Order seen by a: $x\ y\ a\ b\ c\ d\ e$
Order seen by b: $x\ y\ a\ b\ c\ d\ e$
Order seen by c: $a\ b\ c\ d\ e\ x\ y$
Order seen by d: $a\ b\ c\ d\ e\ x\ y$
Order seen by e: $a\ b\ c\ d\ e\ x\ y$

Now suppose each node uses the median select algorithm to select a reference signal to synchronize its own clock. In the above example, this would mean that each node selects the third clock, not counting its own, as the reference signal; that is, nodes a, b, c, d, and e will synchronize to nodes b, a, d, c, and c, respectively. As a result, there are two nonsynchronizing cliques, $\{a, b\}$ and $\{c, d, e\}$. By nonsynchronizing cliques we mean that a and b are synchronized to each other and c, d, and e are synchronized to each other, but there is nothing to prevent a and b from drifting far apart from c, d, and e.

To overcome this problem, Krishna, Shin, and Butler proposed that each node p use the $f_p(N, m)$th clock signal as the reference signal instead of the median signal, where

$$f_p(N,m) = \begin{cases} 2m & \text{if } A_p < N - m \\ m + 1 & \text{if } A_p \geq N - m \end{cases}$$

and where N is the total number of clocks, m is the maximum number of faults to be tolerated, and A_p is the position of node p in the perceived order of the arriving clock signals at node p. They showed that if $N \geq 3m + 1$, then this function for selecting the reference signal will ensure that all the nonfaulty nodes remain synchronized regardless of how the faulty nodes behave.[5]

For the situation above, $N = 7$, $m = 2$, and

$$f_p(7,2) = \begin{cases} 4 & \text{if } A_p < 5 \\ 3 & \text{if } A_p \geq 5 \end{cases}$$

Thus, a synchronizes to b, b synchronizes to c, c synchronizes to e, d synchronizes to e, and e synchronizes to c. This sequence of synchronization among the nodes results in a single clique containing all five nonfaulty nodes.

The problem with Krishna, Shin, and Butler's scheme is that selection of the reference signal is based on the position of the local clock in the perceived sequence

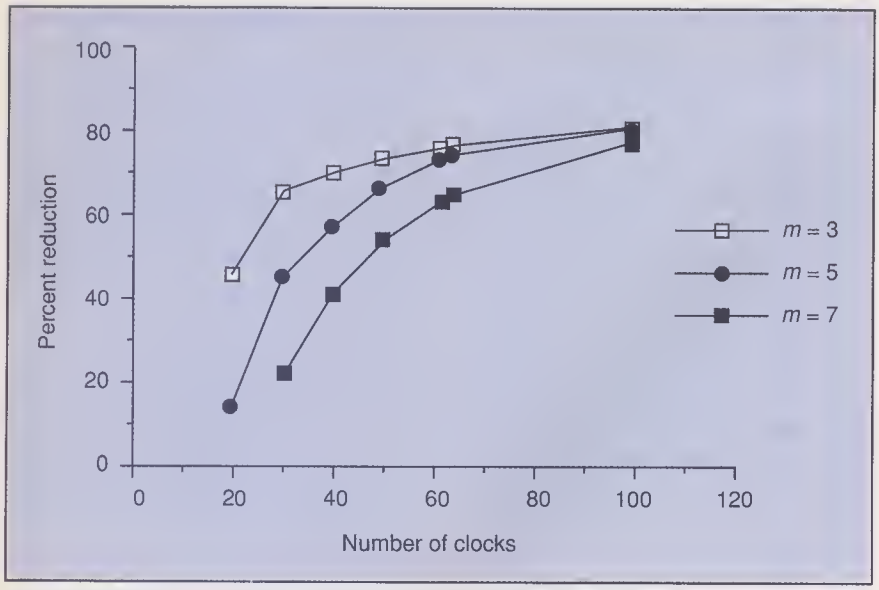


Figure 4. Percentage of reduction over a fully connected network versus system size.

of the incoming clock signals. This complicates the hardware at each node because the hardware must order the incoming clock signals and dynamically choose a reference signal by identifying the local clock's position in the ordered sequence. To overcome this problem, Vasanthavada and Marinos proposed a slight variation of the above scheme, using two reference signals instead of one.[8]

In their scheme it is not necessary to generate the complete sequence of all incoming signals. Instead, it is sufficient to identify the $(m+1)$th and the $(N-m-1)$th clocks in the temporal sequence. This is easy to do in hardware because we can count the clock signals that have arrived and record the identity of the $(m+1)$th and the $(N-m-1)$th clock signals. The average phase difference of the local clock with respect to these two clock signals is used to correct the local clock in the subsequent cycles.

Eliminating major limitations in hardware synchronization. The main advantage of the above hardware synchronization algorithms is that the worst-case skews are several orders of magnitude smaller than those in software synchronization algorithms. However, these hardware synchronization algorithms have two major limitations. The first is that the hardware synchronization algorithms as described above require a fully connected network of clocks. Because of the many interconnections in a fully connected network, this requirement means that the reliability of synchronization would be determined by the failure rates of these interconnections rather than by the failure rate of the clocks. Furthermore, the large number of interconnections would cause problems of fan-in and fan-out.

The second limitation is that the above hardware synchronization algorithms are based on the assumption that the transmission delays are negligible compared with other parameters relevant to the algorithms. In a large system, the physical separation between a pair of clocks can be enough to result in non-negligible transmission delays. A significant difference in the transmission delays between two pairs of nonfaulty clocks can change the order in which a nonfaulty clock perceives other nonfaulty clocks and thus affect the selection of the reference signal.

These two limitations can be eliminated from any of the above algorithms with the help of the recent developments surveyed below.

Interconnection problem. Shin and Ramanathan recently proposed an interconnection strategy for synchronizing a large distributed system using any of the above hardware synchronization algorithms.[6] Their interconnection strategy requires only 20-30 percent of the total number of interconnections in a fully connected network.

The idea behind their strategy is to synchronize the clocks in the system at two different levels. The clocks are first partitioned into several clusters. Each clock then synchronizes itself not only with respect to all the clocks in its own cluster but also with one clock from each of the other clusters. As a result of this mutual coupling between the clusters, the clusters remain synchronized with respect to each other, and the system as a whole remains well synchronized.

Shin and Ramanathan formulate the problem of partitioning the clocks into clusters as a small integer-programming problem. The objective is to minimize the total number of interconnections subject to the constraint that the fault tolerance requirement is satisfied. For each clock, the solution to the integer-programming problem identifies all other clocks that it should monitor. Shin and Ramanathan[6] show that this interconnection strategy does not result in loss of synchronization and the worst-case skew increases by a factor of three as compared with the skew in a fully connected network of clocks.

The reduction in the total number of interconnections over a fully connected network as a function of system size is shown in Figure 4. It follows from this plot that the percentage of reduction increases with system size; this is precisely the situation in which the number of interconnections is a serious problem.

Transmission delay problem. The effects of transmission delay in phase-locked loops have been studied extensively in the communication area but not in the presence of Byzantine faults. Shin and Ramanathan[7] show that it is easy to incorporate ideas from the communication area to take into account the presence of Byzantine faults and non-negligible transmission delays.

Figure 5 illustrates their underlying idea and shows the hardware required at node i to determine the exact phase difference between its clock and the clock at node j. Instead of one phase detector for every node pair, as in other algorithms, this solution requires two phase detectors and an averager. The inputs to the first phase detector, PD_1, are the clock signals from nodes i and j. Since the clock signal from node j encounters a delay in reaching node i, the phase difference detected by PD_1 does not represent the true phase difference between the two clocks.

The inputs to the second phase detector, PD_2, are the clock signals from node j and the clock signal from node i that is returned from node j. In other words, the clock signal

that node j is monitoring is returned to node i to be compared with the signal from node j. Because of the transmission delays, the phase difference detected by PD_2 also does not represent the true phase difference between the two clocks. However, Shin and Ramanathan prove that the average of the outputs of PD_1 and PD_2, e_{ij}, is proportional to the exact phase difference between the clocks at nodes i and j regardless of the transmission delays.

This true phase difference can then be used for any hardware synchronization algorithm. The disadvantages of this approach are the need for two phase detectors instead of one and the doubling of the number of interconnections. However, when this solution is combined with Shin and Ramanathan's interconnection scheme, we get a viable solution even for a very large distributed system.

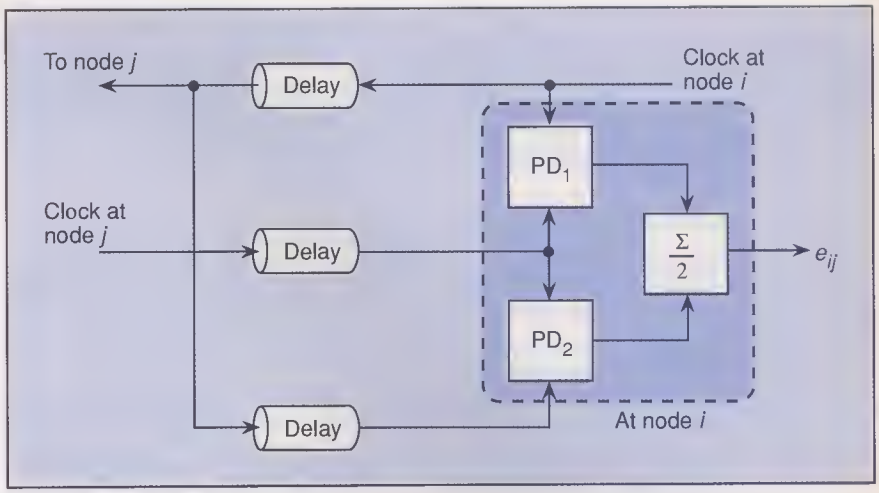


Figure 5. Elimination of transmission delay effects.

Hybrid synchronization

Earlier we noted that the main drawback of software synchronization algorithms is that the worst-case skews are at least as large as the variation in the message transit delay in the system. This can be a serious problem in a large distributed system because this variation can be substantial. For example, in some systems worst-case message transit delays as large as 100 times the mean message transit delay have been observed. In other words, the worst-case skews in such systems would be at least this large if a software clock synchronization algorithm were adopted. At the other extreme, hardware algorithms achieve very tight skews with minimal overhead. The skews in hardware algorithms are typically on the order of tens of nanoseconds,[8] as opposed to tens of milliseconds in the software algorithms. However, the hardware schemes are prohibitively expensive, especially in large distributed systems.

To remove the inherent limitations of both the hardware and software approaches, Ramanathan, Kandlur, and Shin recently proposed a hybrid synchronization scheme that strikes a balance between the clock skews attainable and the hardware requirement.[9] Their scheme is particularly suitable for large, partially connected homogeneous distributed systems with point-to-point interconnection topologies such as hypercubes or meshes.

This hybrid scheme is similar to algorithm CNV.[1] As in CNV, each node has a clock process that is responsible for maintaining the time on that node. At a specified time in the resynchronization interval, a clock process broadcasts the local clock value to all other clock processes. The broadcast algorithm is such that all clock processes receive multiple copies of the clock message through node-disjoint paths. The number of copies used in the broadcast algorithm depends on the maximum number of faults to be tolerated and the fault model for the system. When a clock process receives a clock message sent by some other clock process, it records the time (according to its local clock) at which the message was received. Then, in accordance with the broadcast algorithm, it relays the message to other clock processes.

Before relaying the message, a clock process appends to the message the time elapsed (according to its own clock) since receipt of the message. At the end of the resynchronization interval, it computes the skews between the local clock and the clock of the source node for each copy it has received. The clock process then selects the $(m+1)$th largest value as an estimate of the skew between the two clocks. The average of the estimated skews over all nodes is used as the correction to the local clock. As in CNV, a minimum of $3m+1$ nodes is required to tolerate m Byzantine faults.

Ramanathan, Kandlur, and Shin's algorithm has several advantages over the algorithms discussed earlier. First, the sending of clock values by different nodes occurs throughout the resynchronization interval rather than during the last S time units of the interval. For hypercube and mesh architectures, they show that the resynchronization interval in their algorithm is so large that there is at most one node broadcasting its clock value at any given time. This prevents abrupt degradation in the message delivery times, which occurs when all nodes in the system broadcast clock values almost simultaneously. Second, the only hardware requirement at each node is for time-stamping of clock messages. The third, and probably most important, advantage is that the worst-case skews are about two to three orders of magnitude tighter than the skews in software schemes. Furthermore, because of the hardware support, the worst-case skews are insensitive to variations in message transit delay in the system.

To compare the hybrid synchronization scheme with the various software and hardware synchronization schemes, consider the worst-case skew this algorithm guarantees. Ramanathan, Kandlur, and Shin show that this worst-case skew is

$$\delta = \max\left\{\frac{2(N-m)(\varepsilon+2\rho NU)+2m\varepsilon}{(N-3m)} + \frac{\rho N^2 U}{(N-3m)}, \delta_0+\rho NU\right\}$$

where the notation is the same as that used for characterizing algorithm CNV under "Convergence-averaging algorithms." The slight difference between the above expression and the corresponding expression for algorithm CNV is that for algorithm CNV the read error ε incorporated the effects of message transit delays between any two of the system nodes. This is because algorithm CNV was intended for a fully connected system in which the message transit delays are very small. In con-

trast, the algorithm in the hybrid scheme is intended for a partially connected system in which the message transit delay can be fairly large. Therefore, the effect of message transit delay (U) on the skew has been separated from the effect of other read errors (ε).

The key conclusion we can draw from the above expression is that the worst-case skew is not an explicit function of U as in most software synchronization algorithms but a function of $\rho \cdot U$. As a result, for typical values of $\rho = 10^{-6}$ and $\varepsilon = 20$ microseconds, Ramanathan, Kandlur, and Shin show that the worst-case skew in a 512-node hypercube with $m = 2$ is less than 200 microseconds even when the maximum message transit delays are as large as 50 milliseconds. These skews are still much larger than those achieved by hardware synchronization algorithms, but the cost of hardware synchronization for a system this large would be exorbitant.

Clock synchronization is an important matter in any fault-tolerant distributed system and has been extensively studied in recent years. Unfortunately, the various solutions proposed are difficult to compare because they are presented under different notations and assumptions. Additional difficulty arises because of slight differences in the assumptions made by the different synchronization algorithms. In this article we classified and presented several software and hardware synchronization algorithms as well as a hybrid synchronization algorithm, all using a consistent notation. We also identified the assumptions each algorithm makes.

Software algorithms require nodes to exchange and adjust their individual clock values periodically. Since the clock values are exchanged via message passing, the time overhead these algorithms impose can be substantial, especially if a tight synchronization is desired. They are therefore suitable for applications that can tolerate a loose synchronization between the system nodes.

Hardware algorithms, on the other hand, use special hardware to ensure a very tight synchronization. Although the overhead they impose on the system is minimal, the hardware algorithms are too expensive to use for all but small systems.

The hybrid scheme is cost-effective and achieves a reasonably tight synchronization. It is also suitable for synchronizing large, partially connected systems and hence is the most viable scheme for future distributed systems. ■

Acknowledgments

The work reported in this article was supported in part by NASA under grant No. AG-1-296 and the Office of Naval Research under contract No. N00014-85-K-0531.

References

1. L. Lamport and P.M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults," *J. ACM*, Vol. 32, No. 1, Jan. 1985, pp. 52-78.

2. J.Y. Halpern et al., "Fault-Tolerant Clock Synchronization," *Proc. Third Ann. ACM Symp. Principles of Distributed Computing*, ACM, New York, 1984, pp. 89-102.

3. J. Lundelius-Welch and N. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," *Information and Computation*, Vol. 77, No. 1, 1988, pp. 1-36.

4. T.K. Srikanth and S. Toueg, "Optimal Clock Synchronization," *J. ACM*, Vol. 34, No. 3, July 1987, pp. 626-645.

5. C.M. Krishna, K.G. Shin, and R.W. Butler, "Ensuring Fault Tolerance of Phase-Locked Clocks," *IEEE Trans. Computers*, Vol. C-34, No. 8, Aug. 1985, pp. 752-756.

6. K.G. Shin and P. Ramanathan, "Clock Synchronization of a Large Multiprocessor System in the Presence of Malicious Faults," *IEEE Trans. Computers*, Vol. C-36, No. 1, Jan. 1987, pp. 2-12.

7. K.G. Shin and P. Ramanathan, "Transmission Delays in Hardware Clock Synchronization," *IEEE Trans. Computers*, Vol. C-37, No. 11, Nov. 1988, pp. 1,465-1,467.

8. N. Vasanthavada and P.N. Marinos, "Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Failures," *IEEE Trans. Computers*, Vol. C-37, No. 4, Apr. 1988, pp. 440-448.

9. P. Ramanathan, D.D. Kandlur, and K.G. Shin, "Hardware-Assisted Software Clock Synchronization for Homogeneous Distributed Systems," *IEEE Trans. Computers*, Vol. C-39, No. 4, Apr. 1990, pp. 514-524.

10. F. Cristian, "Probabilistic Clock Synchronization," Tech. Report RJ 6432 (62550), IBM Almaden Research Center, Sept. 1988.

11. F.B. Schneider, "A Paradigm for Reliable Clock Synchronization," Tech. Report TR-86-735, Computer Science Dept., Cornell Univ., Ithaca, N.Y., Feb. 1986.

12. L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems*, Vol. 4, No. 3, July 1982, pp. 382-401.

13. A.L. Hopkins, T.B. Smith, and J.H. Lala, "FTMP — A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," *Proc. IEEE*, Vol. 66, No. 10, Oct. 1978, pp. 1221-1240.

Parameswaran Ramanathan is an assistant professor of electrical and computer engineering at the University of Wisconsin, Madison. From 1984 to 1989 he was a research assistant in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. His activities have focused on fault-tolerant computing, distributed real-time computing, VLSI design, and computer architecture.

Ramanathan received the B. Tech degree from the Indian Institute of Technology, Bombay, India, in 1984 and MSE and PhD degrees from the University of Michigan, Ann Arbor, in 1986 and 1989, respectively. He is a member of the IEEE Computer Society.

Kang G. Shin joined the University of Michigan, Ann Arbor, in 1982, where he is currently a professor of electrical engineering and computer science. In 1985 he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer to validate various architectures and analytic results in the area of distributed real-time computing.

Shin received a BS in electronics engineering from Seoul National University, South Korea, in 1970 and MS and PhD degrees in electrical engineering from Cornell University, Ithaca, New York, in 1976 and 1978, respectively. He is a member of the IEEE Computer Society.

Ricky W. Butler is a research engineer at the Langley Research Center. His research interests lie in the design and validation of fault-tolerant computer systems used for flight-critical applications. He received his BA in mathematics in 1976 and his MS in computer science in 1978, both from the University of Virginia.

Inquiries to the authors can be addressed to Parameswaran Ramanathan, Dept. of Electrical and Computer Engineering, University of Wisconsin, Madison, 3436 Engineering Bldg., 1415 Johnson Dr., Madison, WI 53706-1691.

Do you practice Yourdon/DeMarco Structured Analysis?
Do you need to produce high quality DFD's?
Do you need a tool that fits your budget?

Then MacBubbles™ was made for you!

MacBubbles supports the process of structured analysis with:

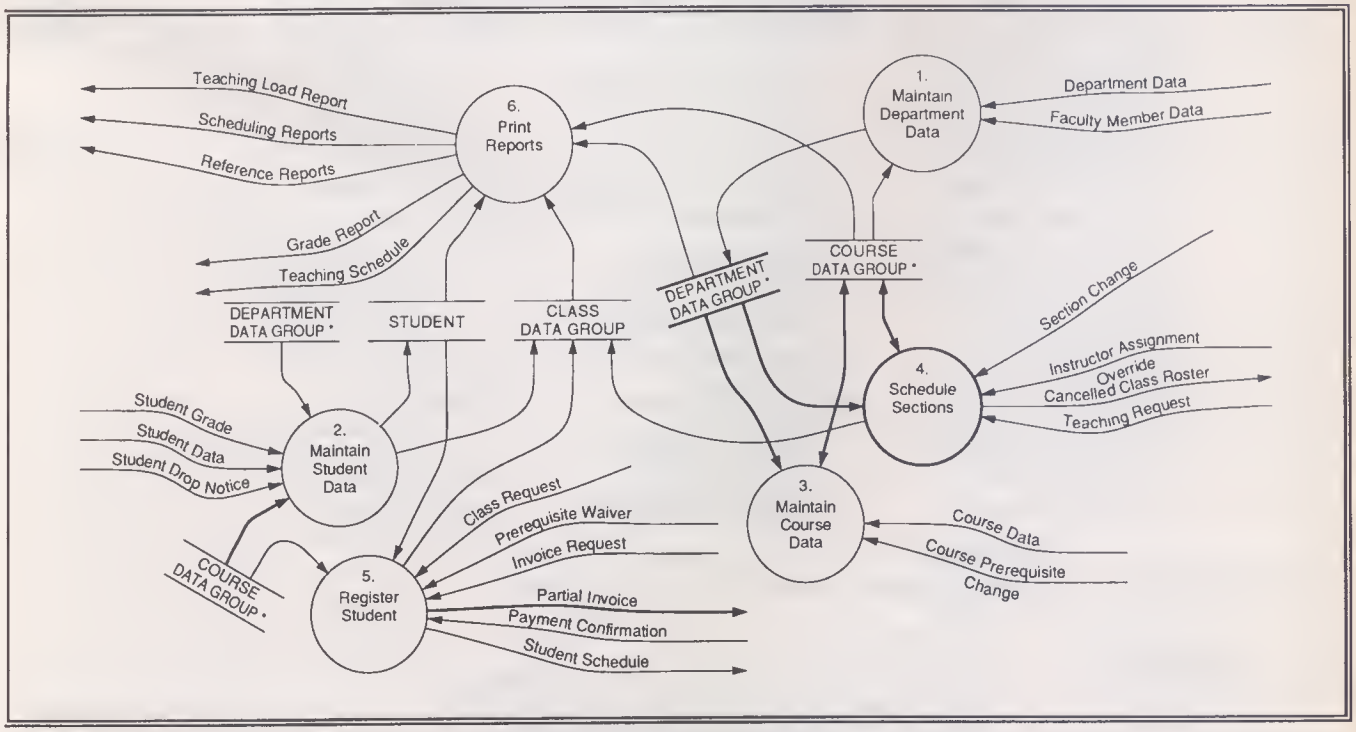
Facilities for creating, modifying, leveling and expanding DFD's
A Data Dictionary that maintains and lists full where-used information
Automated balancing checks for DFD's and minispecs

MacBubbles improves productivity:

Easy to learn
Easy to use
Responsive

MacBubbles is economical:

Single copy price	$779.99
Multi-copy discount	available
Demo disk	$25.00



Sample Data Flow Diagram

MacBubbles requires:

A Macintosh Plus or SE
Two floppy drives or a hard disk
A LaserWriter for high quality output

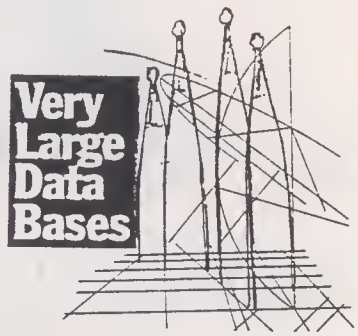
StarSys, Inc.
11113 Norlee Drive
Silver Spring, MD 20902-3619
(301) 946-0522

CALL FOR PAPERS

VLDB'91

17th INTERNATIONAL CONFERENCE ON
VERY LARGE DATA BASES

SEPTEMBER 3-6, 1991
BARCELONA, (CATALONIA, SPAIN)



PAPERS

Original papers of up to 5.000 words in length are invited on topics including, but not limited to, those listed above. All submitted papers will be read and carefully evaluated by the Program Committee. To submit papers, send five (5) copies of double-spaced manuscript, in English, including keywords, to one of the Program Co-Chairs by **FEBRUARY 15, 1991:**

Prof. Dr. Amilcar Sernadas
INESC
Rua Alves Redol, 9 ,7º.
Apartado 10105
P - 1017 LISBOA Codex (Portugal)
Internet (US) : inesc ! acs%solo@relay.EU.net
(Europe) acs%solo@inesc.uucp
or acs@inesc.ctt.pt

Dr. Guy M. Lohman
IBM Almaden Research Center
Dept. K55, Bldg. 801
650 Harry Road
SAN JOSE, CA 95120-6099 (U.S.A.)
Internet: lohman@ibm.com
BITnet: lohman@almaden

S. Abiteboul (France)
P. Apers (Netherlands)
F. Bancilhon (France)
D. Batory (TX, USA)
C. Beeri (Israel)
D. Bitton (CA, USA)
A. Borgida (NJ, USA)
F. Bry (Germany)
J. Bubenko (Sweden)
A. Buchmann (USA)
R. Carapuca (Portugal)
M.Carey (WI, USA)
M.A. Casanova (Brazil)
S. Chakravarthy (FL, USA)
S. Christodoulakis (CDN)
P. Dadam (Germany)
U. Dayal (MA, USA)
D. DeWitt (WI, USA)
K. Dittrich (Switzerland)
H.D.Ehrich (Germany)
E. Falkenberg (Netherlands)
J. L. Fiadeiro (U.K.)
J. Freytag (Germany)
H. Gallaire (France)
H. Garcia-Molina (NJ, USA)
G.Gardarin (France)
T. Haerder (Germany)
R. Hull (CA, USA)
Y. Ioannidis (WI, USA)
H.V. Jagadish (NJ, USA)
Y. Kambayashi (Japan)
R. King (CO, USA)
M.Kitsuregawa (Japan)
R. Krishnamurthy (CA, USA)
P. Larson (CDN)
B. Lindsay (CA, USA)
U. Lipeck (Germany)
F. H. Lochovsky (CDN)
P. Lockemann (Germany)
P. Loucopoulos (U.K.)
H. Lu (Singapore)
P. Lyngbaek (CA, USA)
R. Manthey (Germany)
Y. Masunaga (Japan)
D. McLeod (CA, USA)
R. Meersman (Netherlands)
C. Mohan (CA, USA)
M.Morgenstern (CA, USA)
E. B. Moss (MA, USA)
S. Navathe (FL, USA)
J. M.Nicolas (France)
S. Nishio (Japan)
A. Olivé (Spain)
F. Olken (CA, USA)
Z. M. Ozsoyoglu (OH, USA)
B. Pernici (Italy)
A. Pirotte (Belgium)
R. Ramakrishnan (WI, USA)
N. Roussopoulos (MD, USA)
R. Sacks-Davis (Australia)
G. Schlageter (Germany)
H. J. Schek (Switzerland)
J. Schmidt (Germany)
T. Sellis (MD, USA)
D. Shasha (NY, USA)
A. Solvberg (Norway)
R. Van de Riet (Netherlands)
Y. Vassiliou (Greece)
L. Vieille (France)
C. Zaniolo (TX, USA)

TUTORIALS

Proposals for tutorials should be sent by **January 6, 1991** to:

Prof. Antoni Olivé
Facultat d'Informàtica
Universitat Politècnica de Catalunya
Pau Gargallo, 5
E - 08028 - BARCELONA (Catalonia-Spain)
Ean: olive@lsi.upc.es
From BITnet: olive%lsi.upc.es.@cernvax

PANELS

Proposals for panel sessions should be sent by **March 30, 1991** to:

Prof. Alfonso F. Cárdenas
University of California at Los Ángeles
Dept. of Computer Science
3731 Boelter Hall
LOS ANGELES. CA 90024 (U.S.A.)
Internet: cardenas@cs.ucla.edu

In cooperation with

acm

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Built-in Self-Testing of Random-Access Memories

Manoj Franklin and Kewal K. Saluja
University of Wisconsin-Madison

Built-in self-test techniques are gaining ground in the testing of logic circuits because they offer a cost-effective way to test high-density digital devices. The basic philosophy behind the BIST technique is "let the hardware test itself" — that is, enhance the functionality of a logic circuit to test itself. The BIST concept was first proposed for combinational circuits, but it later found a quick application in the testing of such regular structures as random-access memories, read-only memories, and programmable logic arrays.

In the early days of memory design, test procedures were developed in an ad hoc manner. The fault coverages of these ad hoc test procedures were limited and often indeterminable. This shortcoming, acknowledged by most researchers, motivated the introduction of such fault models as stuck-at faults, decoder faults, coupling faults, and pattern-sensitive faults. By and large, the fault models have been simple. Until recently, researchers did not develop models covering complex cell interactions, because they believed that long tests would be required to detect such faults. In conventional testing environments with external testers, the only tests thought practical for large RAMs were those having a linear relationship with the number of bits, N, in the RAM. Ironically, the larger the RAM, the more complex the fault model required to effectively model the variety of physical failures that could occur because of interference between closely packed cells.

An examination of BIST schemes indicates that approaches based on test architectures rather than on test algorithms are more versatile and will likely predominate in the future.

In a BIST environment, relatively inexpensive testers can perform functional test for testing RAMs. A BIST tester need only power up a chip, initiate the test signal, and read the chip's status. Therefore, much longer tests providing higher fault coverage without excessive cost can be applied. The test time estimates in Table 1 (assuming a 10-megahertz clock) show that for large RAMs $\mathbf{O}(N^2)$-length tests may be unacceptable (an $\mathbf{O}(N^\alpha)$ test means a test of length CN^α, where C is a constant). Nevertheless, $\mathbf{O}(N^{3/2})$-length tests can be practical for memories of 4 megabits or larger, depending on the extent to which the memory's internal organization can be exploited by the BIST logic.

Judging by the current trends, memory size will continue to increase. As memories become larger, BIST becomes more of a necessity because of the high costs incurred by off-line testers for even $\mathbf{O}(N)$ tests. Furthermore, even if the order of test length is moderately high, BIST techniques can bring down the effective test time by using such techniques as parallel testing and line-mode testing.

Another motivation for BIST is that the BIST logic incorporated in a chip can be used for both manufacture testing and in-circuit testing. If the implemented algorithm's test length is sufficiently small, the same BIST logic can even be used for testing RAMs during computer power-on, as part of the CPU's self-test procedures. Current BIST implementations cannot, however, be used for testing when the chip contains useful data.

Although BIST may still be a high-overhead concept (about 20 to 30 percent) for general integrated-circuit designs, it requires

0018-9162/90/1000-0045$01.00 © 1990 IEEE

Table 1. Test time for different test lengths and memory sizes (with a 10-megahertz clock).

Test Length	Memory Size 1 Mbit	4 Mbits	16 Mbits
N	0.1 sec.	0.4 sec.	1.6 sec.
$N \log N$	2.1 sec.	9.2 sec.	40.3 sec.
$N^{3/2}$	1.8 min.	14.3 min.	1.9 hr.
N^2	30.5 hr.	20.3 days	325.8 days

little overhead (less than 1 percent) for application in large RAMs, as a later section shows. Furthermore, recent developments in very large scale integrated circuitry allow moderately complex test algorithms to be built within a chip.

In this context, the article has four major purposes:

- to demonstrate that BIST is a viable solution to the problem of testing large memories,
- to introduce the notion of generic test architectures suitable for implementing a wide variety of test algorithms,
- to provide a taxonomy for test architectures and use this taxonomy to categorize BIST implementations, and
- to survey the important BIST implementations reported by universities and industry.

A fair treatment of these four issues requires a discussion of the fault models and the test algorithms on which the BIST implementations are based.

Fault models for RAMs

Before discussing the important fault models, let's consider how RAM chips are organized. A RAM chip consists of an array of memory cells, an address decoder, address and data registers, and a read/write logic. An N-bit RAM may be organized either as a single-bit output RAM (N-word $\times$ 1-bit RAM) or as a k-bit output RAM (M-word $\times$ k-bit RAM). Generally, an M-word $\times$ k-bit RAM is organized as k identical partitions. Each M-bit partition may itself be organized as $l (l \geq 1)$ two-dimensional arrays of $m \times n$ cells, such that $M = lmn$. Cells and their contents in each of these arrays are independent of the cells in other arrays. By assuming that no interaction can take place between cells of different arrays, we can model faults considering only a two-dimensional array. Therefore, we need only test each of these arrays completely, as opposed to testing the RAM as a single unit. The arrays can be tested sequentially or in parallel if the memory organization permits. The only restriction is that each array must be tested independently of the remaining arrays.

A wide variety of physical failures can occur in the memory array, address decoder, and read/write logic, causing various failures in the memory function. Their causes depend on such factors as component density, circuit layout, and manufacturing method. A number of fault models have been developed to capture the effects of physical failures in RAMs. In this section, we describe the important fault models relevant for the functional testing of RAMs using BIST. Faults not covered include soft faults such as transient faults and intermittent faults. A recent survey paper on fault models and functional testing techniques for RAMs provides more detailed descriptions.[1]

Invariably, two assumptions have been used in the development of all fault models and test algorithms: the single-fault assumption and the nondestructive or fault-free read operations assumption.

The *single-fault assumption* reduces the complexity of test procedures, which become unwieldy for most fault models if the test is designed to detect multiple faults. Tests that detect all single faults often detect most multiple faults. This justifies the use of the single-fault assumption.

The *fault-free reads assumption* has also been used for practical reasons. Test procedures for RAMs often have some form of embedded *checking experiment*, that is, the application of a sequence of writes to bring the memory to a known state and the verification of this state by reading the memory cells. The test procedure becomes extremely complex — and sometimes impossible — if the read operations are assumed to be faulty or destructive. In reality, however, most faults in the read/write logic are easily detected because they result in catastrophic failures.[1] Simple tests can be derived and applied by an external tester in the final testing stages to detect noncatastrophic faults in the read/write logic.

Stuck-at fault model. A memory cell is said to be stuck-at-1 (stuck-at-0) if its contents remain fixed at logic 1 (0), irrespective of what is written into it. Stuck-at faults are also useful for modeling faults in other parts of the memory system, such as the decoder.

Coupling fault model. A pair of memory cells is said to be coupled if a transition in one of them changes the contents of the other cell from 0 to 1 or 1 to 0. Coupling faults are of two types. An *idempotent coupling fault* is one in which a transition in one cell forces the contents of another cell to a certain value (either 0 or 1), whereas an *inversion coupling fault* is one in which the transition causes an inversion in the contents of the second cell. Coupling faults could also exist between three or more cells.

Pattern-sensitive fault model. A memory cell is said to have a pattern-sensitive fault if its state is altered by a pattern of 0's and 1's, $0 \rightarrow 1$ transitions, $1 \rightarrow 0$ transitions, or both $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions in a group of other memory cells. The group of cells that influences the base cell's behavior is called the *neighborhood* of the base cell. The problem of pattern sensitivity arises primarily from the high component densities of RAMs and the related effect of unwanted interacting signals. As RAM density increases, the cells become physically closer, and pattern-sensitive faults become the predominant faults. Moreover, other fault classes that affect the memory cells — shorts, stuck-at faults, and coupling faults — can be regarded as special types of pattern-sensitive faults.

Testing a RAM for unrestricted pattern-sensitive faults is impractical, as it requires an $O(2^N)$ test.[1] This fact has led researchers to consider restricted pattern-sensitive fault models in which the neighborhood size is small. Another restriction is on the positions in the array that a neighborhood is allowed to take. Often the neighborhood is allowed to take only the position that physically surrounds the base cell. Traditionally, the restricted neighborhoods considered are the five-cell and nine-cell physical neighborhoods. Figure 1a shows the five-cell physical neighborhood of a memory cell.

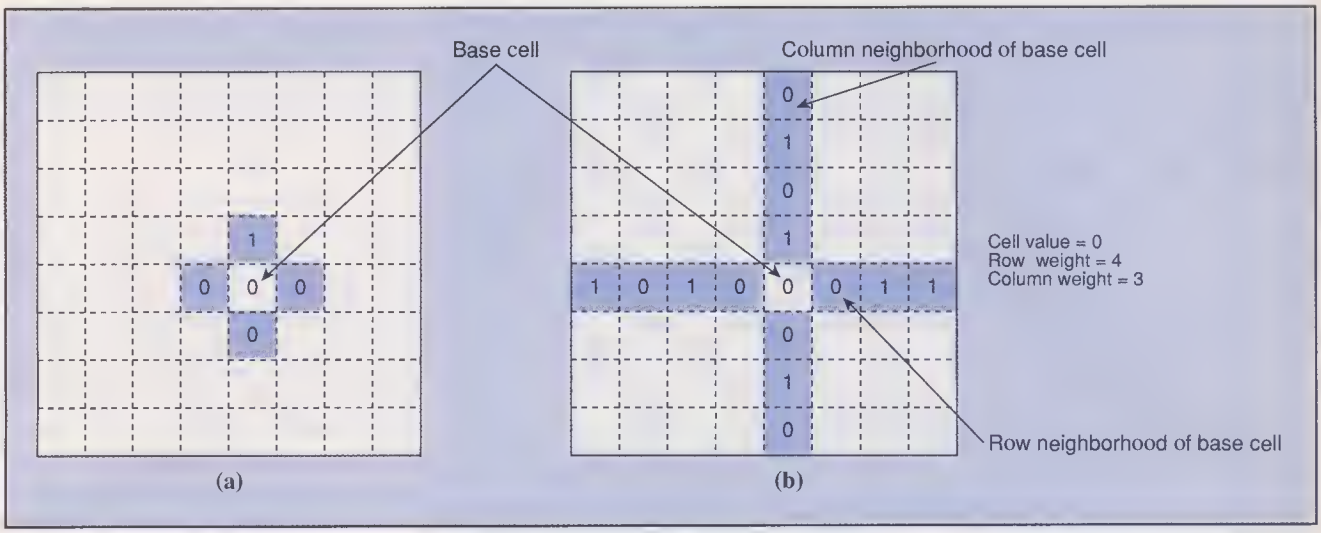


Figure 1. Different types of neighborhood: (a) a five-cell neighborhood; (b) a row/column neighborhood.

Within the context of pattern sensitivity, different fault models have been proposed, based on the type of interaction between the cells. In the *static-pattern-sensitive fault* model, a cell is said to be faulty if its contents change when a certain pattern of 0's and 1's exists in the neighborhood cells (that is, the pattern to which the cell is sensitive is static). A *dynamic-pattern-sensitive fault* is said to occur if the state of a cell changes because of a *change* in its neighborhood pattern. Researchers have also studied variations of these fault models, for example, those using active and passive neighborhoods.[1]

Row/column weight-sensitive fault model. The neighborhoods discussed so far for restricted pattern-sensitive faults are the physical neighborhoods of a cell. The row/column weight-sensitive fault model is based on the broader *row/column neighborhood*.[2] The row (column) neighborhood of a cell consists of all the cells in the same row (column) but excluding the cell. It is related to the electrical neighborhood of a cell because cells of the same row share a common word line, and cells of the same column share a common bit line. The row/column neighborhood is much larger than the conventional five-cell and nine-cell physical neighborhoods described earlier. Row weight of a cell is the number of 1's in its row neighborhood; column weight is the number of 1's in its column neighborhood.

Figure 1b shows the row/column neighborhood, row weight, and column weight of a memory cell. The row/column weight-sensitive fault model is based on the observation that the contents of a cell can be affected by the contents of cells in its row and column neighborhood. Interference could occur between cells of the same column or row, since these cells are electrically connected and share common addressing and refresh circuitry. In the row/column weight-sensitive fault model, a memory cell is said to be faulty if its content is sensitive to any combination of row and column weights.

Besides considering a larger neighborhood than the conventional five-cell and nine-cell neighborhoods, this fault model has an additional advantage: Tests that detect row/column weight-sensitive faults also detect most of the faults modeled by other fault models. Furthermore, weight-sensitive fault tests are also applicable for reconfigurable memory chips, whereas the five-cell-neighborhood pattern-sensitive fault tests are not.

Faults in the decoder and read/write logic. Most faults occurring in the address decoder and the read/write logic can be mapped to faults in the memory cell array; that is, during tests of the memory cell array, they will behave as faults *in* the memory cell array.[1] A stuck-at fault in the read/write logic will appear as a large group of memory cells with a stuck-at fault. Thus, an algorithm that detects stuck-at faults in the memory array can easily detect this fault. The same arguments are valid for coupling faults. Similarly, faults in the address decoder can be modeled by faults in the memory array, so the decoder faults will be detected by tests for the memory cell array.

Test algorithms and their fault coverages

Over the years, several algorithms of different complexities have been developed to test RAMs. The early algorithms were ad hoc; the later algorithms were specifically designed to detect faults from various fault models. Recently, random pattern testing has also been proposed for RAMs. In this article, we discuss only those test algorithms (ad hoc or specific) that have been applied (in original or modified form) for BIST implementation in either universities or industry.

All test algorithms consist of a sequence of writes and reads applied to the cells in the memory array. In our discussion, $W_i \leftarrow v$ denotes the operation "Write value v into cell i." Similarly, $R_i\ (= v)$ denotes the operation "Read cell i, with v as the expected value."

Mscan test. Memory scan is a trivial test procedure developed in an ad hoc manner. The Mscan test writes each cell, first with a 0 and then with a 1. Each value is verified by reading it before a new value is written. The formal algorithm is as follows:

For $i = 0, 1, \ldots, n-1$
$W_i \leftarrow 0$
$R_i\ (= 0)$
$W_i \leftarrow 1$
$R_i\ (= 1)$

The deterministic fault coverage of this test procedure is rather low. All that is known at the end is that there is at least one

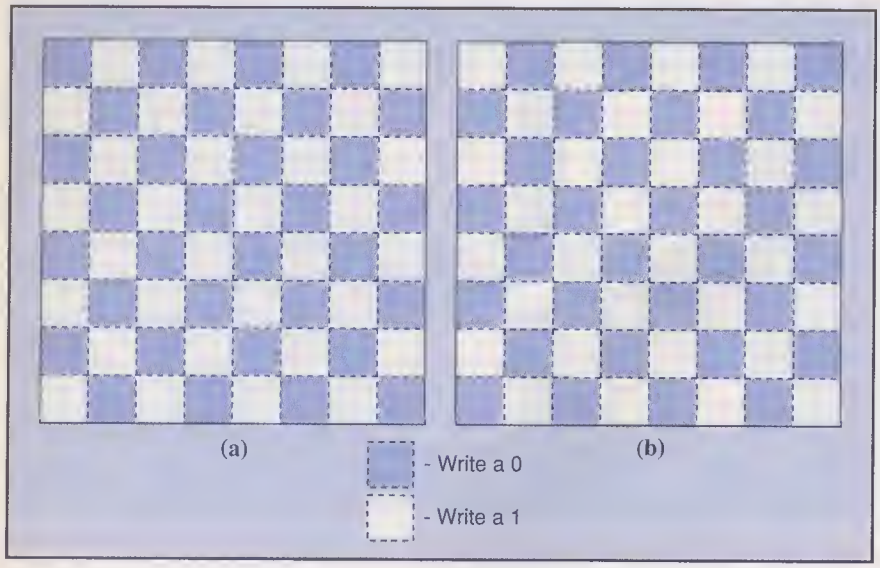


Figure 2. Tiling a memory array for the checkerboard test: (a) pattern one; (b) pattern two.

cell in the RAM that can be set to 0 and 1. This is because a fault in the decoder may cause the same cell to be referenced each time. Since the test performs four operations on each cell, its length is $4N$.

Marching test. Perhaps the most widely used test algorithm in the industry is the marching test. A reason for its popularity is its simplicity, coupled with a moderate fault coverage. The marching-test algorithm initializes the memory array to all 0's, and then scans the memory cells in ascending and descending orders. For each cell, scanning involves reading the cell for the expected value, writing the complement value, and reading it again.

The idea behind this algorithm is that, while scanning the memory in ascending order, any direct coupling between the current cell and a higher address cell is detected when reading the latter. Along with this, any error in the higher address cell due to decoder faults will also be detected. Similarly, scanning the memory in the descending order detects all the effects on lower address cells.

The formal algorithm is given below; the details can be found elsewhere.[1]

Step 1. $W_i \leftarrow 0$ for $i = 0, 1, ..., n-1$

Step 2. For $i = 0, 1, ..., n-1$
$R_i\ (=0)$
$W_i \leftarrow 1$
$R_i\ (=1)$

Step 3. For $i = n-1, n-2, ..., 0$
$R_i\ (=1)$
$W_i \leftarrow 0$
$R_i\ (=0)$

Step 4. Repeat steps 1 through 3, interchanging 0's and 1's.

The marching test detects all stuck-at faults and decoder faults. However, it does not detect all single coupling faults. Different variations of the marching test, all of length $\mathbf{O}(N)$, have been suggested in the literature.[1]

Checkerboard test. This simple algorithm, developed in an ad hoc manner, is designed for two-dimensional memory architectures. The algorithm fills the memory array with a checkerboard pattern by writing 0's and 1's in alternate cells. Two patterns, as shown in Figures 2a and 2b, are written. The cells are read after the application of each checkerboard pattern.

Step 1. $W_{(i,j)} \leftarrow 0$ for $i + j$ = even
$W_{(i,j)} \leftarrow 1$ for $i + j$ = odd

Step 2. $R_{(i,j)}\ (=0)$ for $i + j$ = even
$R_{(i,j)}\ (=1)$ for $i + j$ = odd

Step 3. Repeat steps 1 and 2, interchanging 0's and 1's.

The deterministic fault coverage of this test procedure is rather low. As with the Mscan test, a decoder fault may cause only four cells at most to be referenced. Therefore, all that is known at the end of this procedure is that at least four cells in the RAM can be set to 0 and 1.

Five-cell-neighborhood static-pattern-sensitive fault test. Many algorithms have been proposed to detect five-cell-neighborhood pattern-sensitive faults. All these algorithms are based on tiling the memory array. We briefly explain an algorithm reported by Kinoshita and Saluja, because this algorithm was later implemented as a built-in self-test.[3] Figures 3a and 3b show the tiling arrangements used by this algorithm for the static-pattern-sensitive fault test. The unmarked cells are the base cells. Each base cell is surrounded by four characters (A, B, C, D). The first phase of the test uses the tiling arrangement shown in Figure 3a. During this phase, the base cells are kept fixed at logic 0. The five-cell-neighborhood patterns are applied to the base cells using all four-tuples (16 patterns), consisting of variables A, B, C, and D. The base cells are read after the application of each pattern. The second phase uses the tiling arrangement of Figure 3b, and the above process is repeated. Then both phases are repeated with the base cells at logic 1.

Row/column weight-sensitive fault test. Different test algorithms of varying test lengths have been proposed for testing RAMs for row/column weight-sensitive faults.[2] All the tests are of length $\mathbf{O}(N^{3/2})$ and use divide and conquer by recursive partitioning as the basic strategy. First the border cells of an array are tested. Then the two middle rows and columns are tested, thereby effectively partitioning the array into four, as Figure 4 shows. Partitioning continues recursively until all the cells of the array are tested. Testing the border cells involves scanning the memory array as in the marching test, and this helps to detect decoder faults. The row/column weight-sensitive fault test also detects five-cell-neighborhood pattern-sensitive faults.

Fault coverage. All the algorithms discussed so far have been implemented as built-in self-tests. Some have also been implemented for embedded RAMs in application-specific integrated circuits. Table 2 lists the complexities and fault-detection capabilities of the algorithms. Blank entries indicate that those classes of faults are either not detected or detected only to a small extent. The entries marked "unidirectional" mean that a cell may be sensitive

to one or more patterns or transitions, but all of them change the cell's state from either 0 to 1 or 1 to 0. The table shows that the fault coverages offered by the Mscan test, marching test, and checkerboard test are rather poor.

Test architectures

Generally, test algorithms for RAMs are developed assuming no knowledge of the internal organization of the memory array. This makes sense, because a test algorithm should be generic to be applicable to memories with different internal organizations. Quite often, the internal details of a RAM chip are not released by the vendor and therefore are not available to customers. The BIST logic designer, on the other hand, knows the internal organization and could use this knowledge to reduce the test time and area overhead with possible modifications to the test algorithm that do not sacrifice the fault coverage. For example, for reads and writes the BIST logic may be able to access multiple bits of an array in parallel if the technology and test algorithm allow, instead of accessing the bits serially. Similarly, the internal organization might permit the testing of multiple arrays in parallel. The modifications made to test algorithms to suit memory's internal structure are analogous to the modifications made to high-level-language computer programs by optimizing and vectorizing compilers that take advantage of the computer's internal organization.

So far, BIST logic design has been driven in an ad hoc manner by the desire to implement specific test algorithms. That

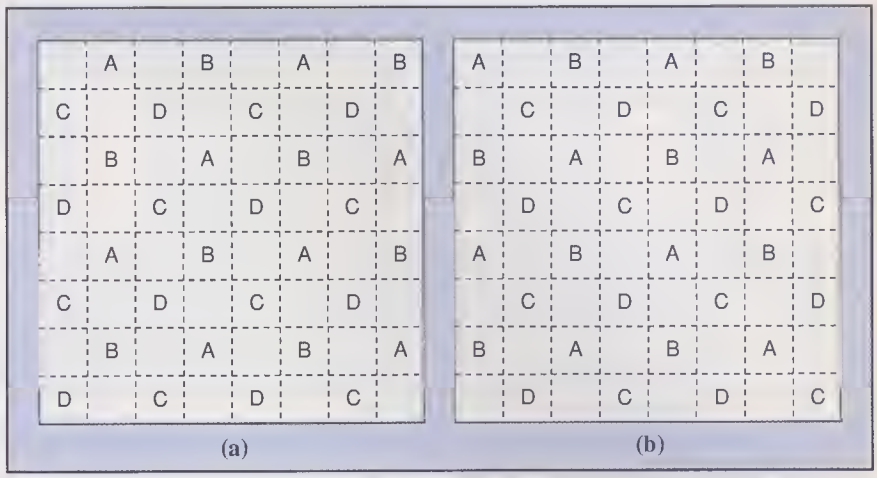


Figure 3. Tiling a memory array for the static-pattern-sensitive fault test: (a) phase one; (b) phase two.

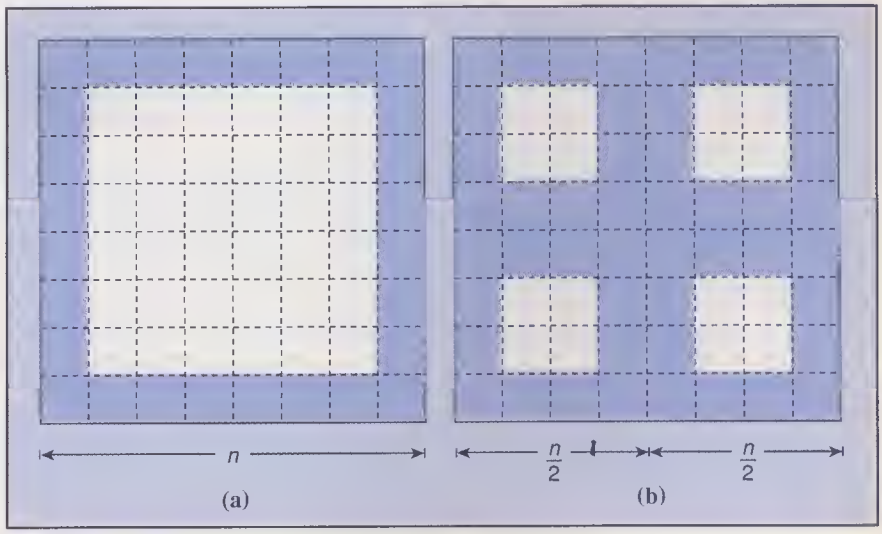


Figure 4. Partitioning a memory array into four in the row/column weight-sensitive fault test: (a) memory array with the border cells tested; (b) memory array partitioned into four.

Table 2. Summary of faults detected by the test algorithms.

Test Procedure	Order of Test Length	Detected Faults			
		Stuck-at-Faults	Coupling Faults	Restricted PSF	Row/Column WSF
Mscan Test	$O(N)$	Does not detect decoder faults			
Marching Test	$O(N)$	All	Does not detect all single coupling faults		
Checkerboard	$O(N)$	Does not detect decoder faults			
SPSF Tests	O(N)	All		Unidirectional	
Row/Column Test	$O(N^{3/2})$	All	Unidirectional	Unidirectional	Unidirectional

Mscan - memory scan
PSF - pattern-sensitive faults
SPSF - static-pattern-sensitive fault
WSF - weight-sensitive faults

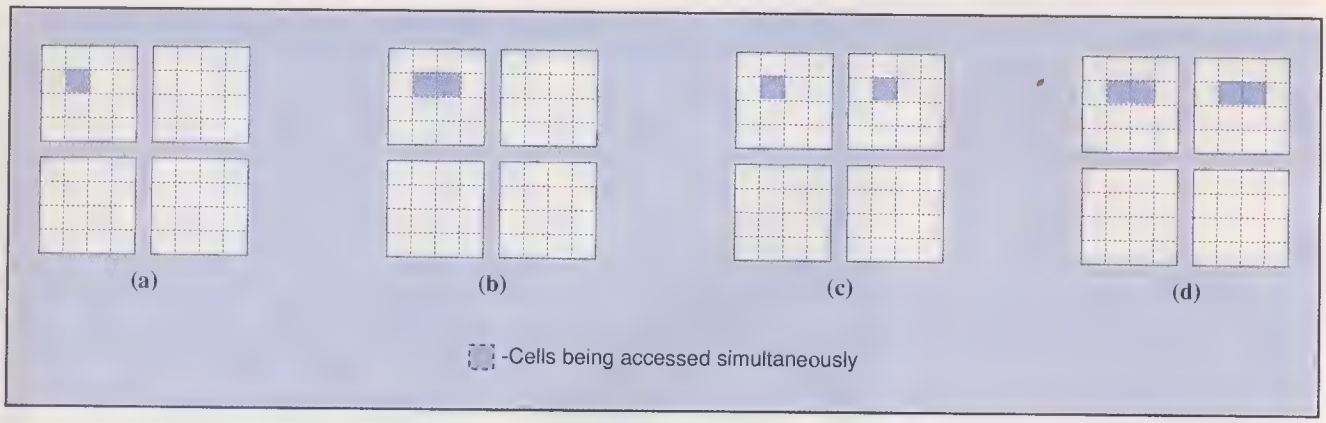


Figure 5. Memory cell accesses for (a) single-array single-bit, (b) single-array multiple-bit, (c) multiple-array single-bit, and (d) multiple-array multiple-bit architectures.

is, a new BIST logic design is carried out for each new algorithm implemented. Much design time goes into the implementation of features common to all test algorithms. This ad hoc approach also makes it difficult to integrate a number of test algorithms on the same chip. An alternative (structured) approach is to develop generic test architectures for implementing a number of test algorithms. For example, in a microcode-based test architecture, it might be possible to implement a class of different test algorithms by changing the microcode, just like changing the microarchitecture of a computer.* The test architecture proposed by Matsuda et al. seems to be a step in the right direction.[4]

Taxonomy. Although a number of RAMs have been implemented with BIST functions, to the best of our knowledge, no one has classified these implementations. Here we provide a taxonomy for classifying BIST RAM test architectures. The categories in a rigorous taxonomy must be exclusive to avoid ambiguity and exhaustive to avoid incompleteness, providing an unambiguous category for every instance presented to it. Our taxonomy matches the number of simultaneously tested arrays and the number of simultaneously accessed bits within an array. Accordingly, we can classify all BIST RAM implementations into one of four test architectures:

- single-array single-bit,
- single-array multiple-bit,
- multiple-array single-bit, and
- multiple-array multiple-bit.

Single-array single-bit (SASB) test architectures are those in which a single array of the RAM chip is tested at a time and a single bit of the tested array is accessed at a time. Since a maximum of one bit from the entire memory chip is accessed at any instant, SASB architectures require the maximum amount of time for testing. Some classes of faults, such as arbitrary coupling faults, restrict the choice of test architecture to SASB architectures. Before the introduction of design for testability and BIST, external tester-based testing also limited the choice mostly to SASB architectures, because only one address can be transmitted from the tester to the chip at a time.

Single-array multiple-bit (SAMB) architectures test a single array at a time, but within the tested array, multiple bits are accessed simultaneously. Generally, the accessed multiple bits are all from the same row; multiple cells from the same column are not accessed simultaneously, as this slows memory access. Multiple bits can be accessed by modifying the address decoder. An SAMB test architecture in which all the n cells of a row (word) within an array are accessed simultaneously has often been referred to as *line-mode testing*.

Multiple-array single-bit (MASB) test architectures can be used if a memory chip is organized as a number of independent arrays, allowing multiple arrays to be tested simultaneously. A single bit from each array is accessed at a time. The concept is similar to the simultaneous testing of many memory chips using external test equipment. In the MASB architecture, a maximum of kl cells can be accessed at a time, where kl is the number of arrays in the memory chip.

Multiple-array multiple-bit (MAMB) architectures use a combination of multiple-array and multiple-bit testing. A number of arrays are tested simultaneously, with a number of cells (normally within a row) in each array accessed simultaneously. Therefore, as many as kln cells can be accessed simultaneously. Sridhar's parallel simultaneous testing of a number of bits from *all* the arrays is an example.[5]

Figure 5 illustrates these concepts with a RAM organized as four arrays. The above discussion demonstrates that the SAMB, MASB, and MAMB architectures provide a speedup over the SASB architecture. An SAMB architecture can, at best, reduce the test length by a factor of n, if all the n cells in a row within an array are accessed simultaneously. The effective speedup can be less than n for certain classes of test algorithms because, during some stages of testing, these algorithms require the contents of part of the row to be kept unchanged when the rest of the row is being tested. Examples are the ping-pong test for coupling faults[6] and the row/column weight-sensitive fault test.[2] The MASB and MAMB architectures can give a maximum speedup of kl and kln, respectively.

Some algorithms are inherently serial and therefore do not attain the maximum speedup offered by the test architecture. Given a test algorithm, the BIST designer must choose one of the four architectures, considering test time, speedup, and technology. Alternatively, given a memory chip design, the BIST designer can select a test architecture based on the available technology and silicon area and then select test algorithms that can be implemented on the

*This concept is similar to the idea of developing general-purpose computers (architectures) to do a variety of computations, as opposed to developing a special-purpose computer for each computation.

selected architecture. Such an approach uses more efficiently the silicon area set aside by the memory designers for the test logic. Only algorithms with good fault coverage should be selected.

All the test implementations reported so far can be categorized into one of the above four test architectures. Memory sizes have now reached the stage where an SASB architecture is almost impractical. Generally, as the memory size increases, the number of arrays increases, with the size of an array remaining more or less constant. Therefore, in the future we can expect many more designers to use the MASB and MAMB test architectures.

Modifying test algorithms. If the arrays are independent of each other and cells of different arrays do not interact, an algorithm developed for an SASB architecture need not be modified for a MASB architecture. However, modifications may be required to implement a conventional SASB algorithm in an SAMB or MAMB architecture.

Most test algorithms can be modified to benefit from the simultaneous access of multiple bits of a row. When multiple bits of an array are accessed simultaneously, faults due to interactions between the simultaneously accessed cells may not be detected, unless special care is taken. Sridhar describes a method to detect errors caused by interactions between cells accessed simultaneously.[5]

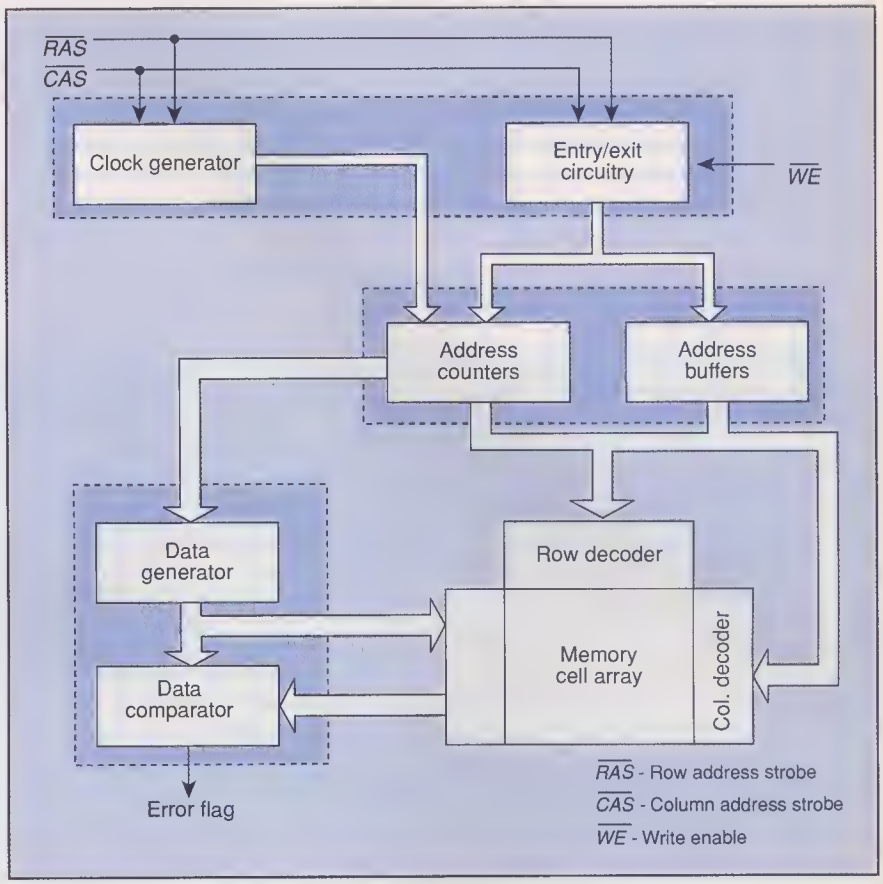


Figure 6. A generic block diagram of random-logic-based BIST logic for RAMs (based on Ohsawa et al.[7]).

BIST logic

Memory chip designers generally use aggressive design rules to maximize the number of cells in a chip and to minimize the memory access time. This imposes rather hard constraints on the BIST logic designer. In general, the BIST logic designer tries to minimize

- the area occupied by the BIST hardware,
- the performance penalty incurred for the normal memory operation,
- the number of additional pins required,
- the disparity between the functional speed and testing speed, and
- the test time, by using the memory's internal structure.

Conceptually, the BIST logic can be divided into four parts: control logic, address-generation logic, data-generation and response-verification logic, and test-trigger logic. Figure 6 (based on Ohsawa et al.[7]) shows a generic BIST organization for testing RAMs. We have modified the figure to show the main parts of the BIST logic clearly.

Control logic. The control logic initiates and stops testing and supervises the control flow of the test algorithm. It can be implemented using random logic or microcode. Random logic offers higher speed and has traditionally been used for designing the control logic; nevertheless, recent designs seem to prefer microcode-based control. For large memories (4 megabits and more), microcode-based BIST design has been shown to have an area overhead which does not exceed that of random-logic-based designs.[3] Therefore, the flexibility and implementational ease offered by microcode makes it superior to random logic for large RAMs.

Microcode fits well with RAM technology because of its regular structure. For BIST RAMs, it may be even more area efficient than random logic, because the aggressive design rules used for RAM cells can also be used for the microcode array. Furthermore, the designer can use such microcode-optimization techniques as microprocedures, microstacks, and encoding by grouping of microinstruction fields, developed for microcode-based computers.

Address-generation logic. Almost all test algorithms require the addresses to be generated in a fairly uniform manner. The control logic can be designed to generate the addresses, but leaving this task to a separate unit is better. For most algorithms, address generation can be achieved by linear-feedback shift registers, registers, or counters, with occasional intervention from the control logic. With the MASB and MAMB architectures, a single address-generation unit can be used for testing multiple arrays.

Data-generation and response-verification logic. The data-generation unit produces the test pattern(s) to be written in the cells. Given a test architecture, differ-

ent strategies can be used for data generation as well as response verification. Linear-feedback shift registers or counters assisted by the control logic can generate data. In an SASB architecture, the correctness of the read values can be verified either by comparing them against the expected values or by signature analysis. Direct comparison is superior, because it can locate single stuck-at faults. Furthermore, with signature analysis, some faults may go undetected because of aliasing errors. For the SAMB, MASB, and MAMB architectures, other fault-detection methods — in addition to comparison against expected values — are comparison of values read from multiple bits, AND reading, and OR reading.

In the MASB and MAMB architectures another convenient verification method is comparing the outputs of symmetrically

BIST implementations

Below we examine several academic and commercial BIST implementations. Our purpose is not a definitive comparison; rather, it is to discuss important implementation features such as test algorithms, test architectures, and control logic. We have tried to include most of the BIST implementations reported in the literature and to present them more or less in chronological order. The terminology is that used by the authors.

On-chip compact test scheme. Saluja, Sng, and Kinoshita proposed a BIST design for testing RAMs for five-cell-neighborhood pattern-sensitive faults.[1] They grouped the patterns required to test a cell for such faults into different classes, depending on the "weight" of the patterns. Then they developed procedures to apply these patterns using optimal test length sequences. The grouping of the patterns helped to keep the test generators and response verifiers simple for BIST implementation, without causing information loss due to data compaction at the response-verification end.

For the control logic, they proposed both random-logic-based and microcode-based designs. They also proposed a simple and effective instruction set for the microcode-based implementation. The on-chip compact test scheme was proposed and studied in an SASB architecture, although the feasibility of using it in a MASB environment was considered. Estimated area overhead for a 64-kilobit static RAM is 1.21 percent; for a 1-megabit static RAM, 0.09 percent.

Parallel test using signature analyzer. Sridhar proposed a scheme that uses a parallel signature analyzer to access bit lines from different arrays simultaneously.[2] The parallel signature analyzer operates in three modes: the scan mode, the write mode, and the signature/read mode. In the scan mode, the analyzer is loaded with a specific pattern from outside the chip by serially scanning in the bits. This mode is also used for scanning out the contents of the analyzer (the signature) at the end of the test. In the write mode, the value stored in the analyzer is written to a number of bit lines in parallel. Finally, in the signature mode, the contents of the memory cells written earlier are read, and a new k-bit signature is generated. This signature determines whether an error has occurred.

This is not a fully BIST scheme, because it requires the scanning in of data from outside the chip. The scheme uses the marching-test algorithm, modified to suit parallel testing. It can be categorized as a MAMB approach, as the parallel signature analyzer can access multiple bits from multiple arrays simultaneously. The MAMB architecture results in very fast testing.

A potential problem with the scheme is that it requires an external tester to scan in the data. Another problem is the low fault coverage offered by the marching test. If a wide enough parallel signature analyzer is used, then the probability for aliasing errors will be very low. The error-detection capability can be significantly enhanced by monitoring the quotient bit of the analyzer, in addition to verifying the final signature. Estimated area overhead for a 256-kilobit dynamic RAM is 1.0 to 2.2 percent; for a 64-kilobit static RAM, 1.8 to 2.9 percent.

Self-testing dynamic RAM. You and Hayes proposed another type of parallel testing: reconfiguring the cells of an array into a circular shift register and using a built-in test generator to test multiple bits concurrently.[3] The dynamic RAM is organized as two identical arrays, and the arrays are tested in parallel to reduce the overall testing time. The test patterns are generated by reconfiguring the cells of each array to act as a circular shift register during testing. When a row (that is, a word line) within an array is activated, the contents of the n cells of the row are transferred to n bit lines, sensed by n sense amplifiers, and then written to the adjacent cells in the same row. Each n-cell row thus acts as an n-bit shift register. The data shifted out from the right-most sense amplifier is saved in a flip-flop and is stored in the left-most cell of the next row in the next shift cycle.

Thus, all m rows of an array effectively form an mn-bit shift register. By saving the initial contents of the right-most cell of the last row, the array realizes an mn-bit circular shift register. The standard sense amplifier circuits are modified so that when a bit value is read from one cell, it can be written into the adjacent cell in the same row. This is accomplished by introducing pass transistors between the physically adjacent bit lines and may adversely affect the sensitivities of the sense amplifiers and the RAM access time in the normal operation mode.

An on-chip comparison circuit consisting of exclusive-OR gates detects faults by comparing the outputs of symmetrically placed cells of the two arrays. The self-testing dynamic RAM implementation can be categorized as a MAMB test architecture, since two arrays are tested in parallel and multiple bits of an array (all the bits of a row) are accessed simultaneously. This scheme detects bit-line imbalance faults and restricted types of pattern-sensitive faults in which a write operation becomes faulty in the presence of a few specific patterns in the cell's adjacent cells. It does not detect faults caused by transitions in the neighborhood. The area overhead for a 4-kilobit dynamic RAM is about 12 percent, and the estimated overhead for a 1-megabit dynamic RAM is about 5 percent.

Parallel testing for VLSI memories. Inoue et al. proposed the line-mode test, a special case of SAMB testing.[4] In the line-mode test, all cells connected to a word line are tested simultaneously. The on-chip test circuit can perform parallel write and parallel compare. The parallel write circuit writes data into all cells connected to a word line, and the parallel compare circuit compares the data in parallel with the expected data. Apart from the memory cell arrays, separate tests check the decoders, the test logic, and the I/O circuits. The memory cells are tested with the marching-test algorithm.

The test circuit occupies less than 1 percent of the chip area for a 2-megabit dynamic RAM. The parallel write operation allows only certain patterns to be applied to the cells; therefore, the technique

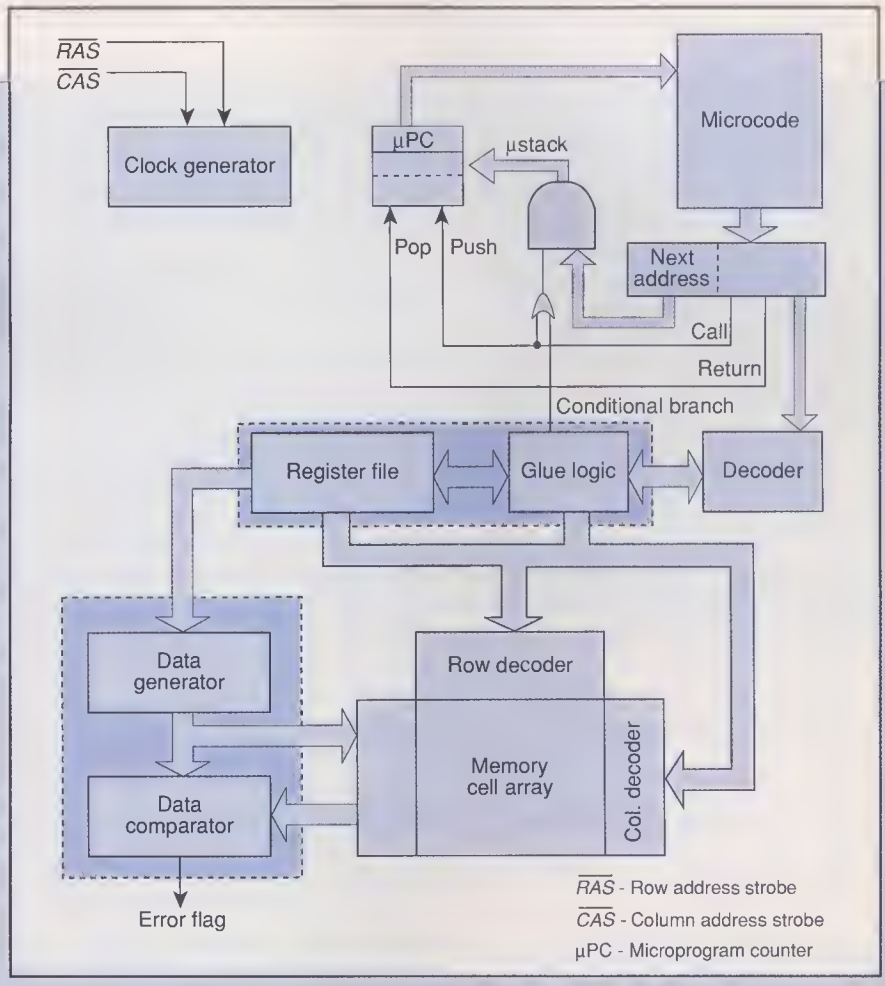
placed bits in the tested arrays. An advantage of the parallel comparison methods is that the expected values need not be generated. A basic assumption is that all bits would not simultaneously have erroneous values. In large dynamic RAMs, memory cell pitches are very small. Thus, the additional area of parallel write and parallel compare circuits should be small enough to be arranged in the small pitch.

Test trigger logic. All BIST RAMs have a normal mode in which the BIST logic is inactive and one or more test modes in which the BIST logic is active. The test modes can be entered using overvoltages, extra package pins, or unique timing sequences with such inputs as Chip Enable, Write Enable, Row Address Strobe, and cannot clearly identify interference between memory cells.

Parallel testing for pattern-sensitive faults. Mazumder and Patel proposed a BIST parallel testing scheme in which a number of cells on the same word line are accessed simultaneously.[5] The decoder is modified so that in the test mode multiple bit lines are selected, allowing the same data to be simultaneously written to multiple cells of the same word line. In the read mode, a multibit comparator concurrently compares the outputs of the bit lines. The additional hardware is designed to fit within the intercell pitch. The algorithm detects both static- and dynamic-pattern-sensitive faults over the nine-cell neighborhood of every cell. Mazumder and Patel estimate the area overhead for a 256-kilobit RAM to be about 0.4 percent.

Built-in processor for self-test. Ritter and Muller have reported a BIST scheme in which a built-in processor tests and repairs large RAMs.[6] Repair requires fault localization and computation of a repair plan, calling for an intelligent self-test concept. This and the demand for high flexibility necessitated a test processor. The test processor also allows for easier and faster adaptation to various types of memory technology and organization. Furthermore, complex algorithms can be incorporated later, when new memory technologies are developed or new types of faults and fault models (not yet considered) are introduced during product life. The test function can be applied not only at manufacturing time, but also during incoming inspection and system-maintenance service. The main components of the test processor are RAM cells, ROM cells, and decoders. The size of the ROM holding the test program is 512 × 14 bits. The area overhead for a 1-megabit RAM is about 5 percent, and the test architecture can be classified as an SAMB architecture.

CMOS dynamic RAM with BIST function. Perhaps the first fully BIST implementation in the industry was the one reported by Ohsawa et al. for a 4-megabit RAM.[7] Their scheme implements a checkerboard test pattern and its complement, and their test architecture falls under the MAMB category. The RAM is divided into eight arrays, two of which are activated in a read/write cycle. From each of the tested arrays, eight bits are accessed simultaneously. A data comparator compares the read pattern with the expected pattern. The control logic is implemented by random logic. The BIST mode is entered through a unique timing sequence. Figure 6 in the main text shows a block diagram of this scheme. The area overhead for the BIST logic is less than 1 percent. A potential problem is the low fault coverage of the checkerboard test.



Block diagram of microcode-based BIST logic for the row/column weight-sensitive fault test.

Row/column pattern-sensitive fault test implementation. The row/column weight-sensitive fault test algorithm has been implemented using both random-logic-based and microcode-based designs.[8] Both schemes use the SASB architecture. We shall briefly describe the microcode-based implementation to provide more insight into the workings of microcode-based BIST. The figure above shows a simplified block diagram of the microcode-based implementation. The main control flow is implemented using microcode, and such support functions as the address-generation logic are implemented using registers and combinational logic. Central to the BIST logic is the 15 ×

Column Address Strobe. Using unique timing sequences is better than using overvoltages and extra package pins, because the latter methods may be incompatible with existing systems. Also, the overvoltage method requires either an additional power supply or the generation of an additional voltage signal. Voss et al. describe an implementation of a 256-kilobit × 1-bit static RAM with multiple test modes in which the test modes are entered by a unique timing sequence.[8] They also describe how a particular test mode can be selected using the normal address input pins, if there are multiple test modes. Miyaji et al. describe the design and implementation of a test trigger circuit for megabit static RAMs that uses the Chip Enable and Write Enable signals to generate a unique timing sequence for entering the test mode.[9]

10-bit control store, which controls the initialization, sequencing, and completion of testing. The control store is conceptually divided into four microroutines. Control passes from one microroutine to another when the former issues a call signal to the latter; control passes back to the former when the latter issues a return signal. A microstack stores the return addresses in the proper order during nested calls. A 4-bit microprogram counter points to the microinstruction currently being executed.

The address-generation logic consists of a register file and some combinational logic (glue logic). The registers hold the row and column addresses of the cell being tested and the cell being read or written. The width of the registers depends on the memory organization and the size of the cell array. The microcode initializes and updates the register file.

A major innovation of this scheme is the implementation of a moderately complex algorithm with a small control store, using microcode-optimizing techniques such as microprocedures and microstacks. The row/column weight-sensitive fault test has higher fault coverage than the other algorithms (see table below). A potential problem with the SASB implementation is that the test time is comparatively long. However, the test time can be reduced by using the MASB or MAMB test architectures. The area overhead of the random logic design for a 4-megabit RAM is less than 0.8 percent.

Serial interfacing for embedded-memory testing. Nadeau-Dostie, Silburt, and Agarwal proposed a serial interfacing scheme for testing embedded RAMs.[9] Embedded RAMs are on-chip RAMs

Summary of different implementations. (Fault coverage for each implementation can be inferred from Table 2 in the main text, which shows the coverage for each type of algorithm.)

Implementation	Algorithm	Test Architecture	Control Logic	Type of RAM
On-chip compact test scheme	SPSF test	SASB	Random logic microcode	SRAM
Parallel test using signature analyzer	Marching test	MAMB	Random logic	SRAM, DRAM
Self-testing DRAM	Restricted PSF test	MAMB	Random logic	DRAM
Parallel test for VLSI memories	Marching test	SAMB	Random logic	DRAM
Parallel test for PSFs	PSF test	SAMB	Random logic	DRAM
Built-in processor for self-test	Not specified	SAMB	Processor	
CMOS DRAM with BIST	Checkerboard test	MAMB	Random logic	DRAM
Row/column test implementation	Row/column test	SASB	Random logic microcode	SRAM
Embedded-memory testing	Marching test Galpat Walk	MAMB	Random logic	SRAM
16-Mbit CMOS DRAM	Marching test Mscan test	SAMB	Microcode	DRAM

MAMB - multiple-array multiple-bit
SAMB - single-array multiple-bit
SASB - single-array single-bit
SPSF - static-pattern-sensitive fault
PSF - pattern-sensitive fault

Future trends

BIST technology combines several different areas: fault models, test algorithms, test implementation, and fault diagnosis. Changes can be expected in each of these areas as technology progresses. Below, we describe how these areas are likely to be affected.

Fault models. State-of-the-art memory chips are designed with spare rows and columns meant for reconfiguration. During manufacture, the memory is tested and repaired (if necessary) by bringing in the spare rows and columns. With such new fault-tolerance techniques as dynamic reconfiguration, fault models based on logical adjacency become irrelevant, whereas those based on physical adjacency and electrical connectivity become more relevant. Future fault models must consider the effects of such reconfiguration within memory chips. With the new high-speed RAM realizations, such as gallium arsenide RAMs, future models must also consider delay faults.

Test algorithms. When a memory chip is reconfigured, physically adjacent cells may no longer have consecutive addresses. Test algorithms for the detection of physical neighborhood pattern-sensitive faults have to account for this fact. Furthermore, we believe that there will be a trend to find optimal or near-optimal, yet simple, test algorithms. These will save testing time as memory size continues to grow, and BIST logic will be used for maintenance testing of RAMs embedded in a system.

Test implementation. When a memory chip is being used in a system (that is, when the chip contains valid data), it cannot be tested on line because the test procedure might destroy the memory contents. Future systems may implement BIST algorithms that have on-line test capabilities. Further research is required not only to develop such algorithms, but also to determine the merits and demerits of such an approach, especially since most memory systems use error-correction code at some level.

Fault diagnosis and self-reconfiguration. In general, current BIST implementations cannot diagnose faults. In the future, BIST will potentially be used in field diagnosis. Such diagnosis will help in reconfiguration of memory chips and repair of multichip memory modules (silicon mass storages).

The separation of test algorithm and test architecture clearly shows the range of possible implementations for a given test algorithm. The test architecture-based approach is more versatile than the ad hoc design approach for BIST logic design, especially with various design constraints. It also facilitates the integration of a number of test algorithms within the same chip.

Our taxonomy for classifying BIST ar-

whose address, data, and read/write controls cannot be directly controlled or observed through the chip's I/O pins, making them good targets for BIST applications. The implemented scheme involves shifting data from one memory cell to another, similar to the self-testing dynamic RAM method described earlier. Although both schemes use the MAMB architecture, there are some differences. While the self-testing dynamic RAM scheme shifted data only within an array and independently tested two arrays in parallel, the new scheme shifts data within an array as well as across arrays, by shifting the data at the end of one array to the beginning of next array in a daisy-chained fashion. This makes sharing the BIST logic among multiple arrays easier, because fewer interconnection lines need to be routed between the BIST logic and the RAM blocks. Furthermore, in the serial interfacing scheme, multiplexers implement the shifting along the I/O data path. Therefore, no modification is required in the RAM. The implemented algorithms are adaptations of the marching test, Galpat (galloping patterns), and walk algorithms.

16-Mbit CMOS DRAM with BIST function. Using microcode-based control logic, Takeshima et al. have implemented the marching test and a scan read/write test with a checkerboard pattern for a 16-megabit dynamic RAM.[10] The size of the control store is 18 × 10 bits. Perhaps theirs is the first industrial BIST RAM implementation using microcode. The dynamic RAM enters the test mode through a unique timing sequence.

References

1. K.K. Saluja, S.H. Sng, and K. Kinoshita, "Built-In Self-Testing RAM: A Practical Alternative," *IEEE Design & Test of Computers*, Vol 4, No. 1, Feb. 1987, pp. 42-51.
2. T. Sridhar "New Parallel Test Approach for Large Memories," *Proc. Int'l Test Conf.*, Computer Society Press, Los Alamitos, Calif., Order No. 641 (microfiche only), 1985, pp. 462-470.
3. Y. You and J.P. Hayes, "A Self-Testing Dynamic RAM Chip," *IEEE J. Solid-State Circuits*, Vol. 20, No. 1, Feb. 1985, pp. 428-435.
4. J. Inoue et al., "Parallel Testing Technology for VLSI Memories," *Proc. Int'l Test Conf.*, Computer Society Press, Los Alamitos, Calif., Order No. 798 (microfiche only), 1987, pp. 1,066-1,071.
5. P. Mazumder and J.H. Patel, "Parallel Testing for Pattern-Sensitive Faults in Semiconductor Random-Access Memories," *IEEE Trans. Computers*, Vol. 38, No. 3, Mar. 1989, pp. 394-407.
6. H.C. Ritter and B. Muller, "Built-In Test Processor for Self-Testing Repairable Random Access Memories," *Proc. Int'l Test Conf.*, Computer Society Press, Los Alamitos, Calif., Order No. 798 (microfiche only), 1987, pp. 1,078-1,084.
7. T. Ohsawa et al., "A 60-ns 4-Mbit CMOS DRAM with Built-In Self-Test Function," *IEEE J. Solid-State Circuits*, Vol. 22, No. 5, Oct. 1987, pp. 663-668.
8. M. Franklin, K.K. Saluja, and K. Kinoshita "Built-In Self-Test Algorithm for Row/Column Pattern Sensitive Faults in RAMs," *IEEE J. Solid-State Circuits*, Vol. 25, No. 2, Apr. 1990, pp. 514-524.
9. B. Nadeau-Dostie, A. Silburt, and V.K. Agarwal, "Serial Interfacing for Embedded-Memory Testing," *IEEE Design & Test of Computers*, Vol. 7, No. 2, Apr. 1990, pp. 52-63.
10. T. Takeshima et al., "A 55-ns 16-Mb DRAM with Built-in Self-Test Function Using Microprogram ROM," *IEEE J. Solid-State Circuits*, Vol. 25, No. 4, Aug. 1990, pp. 903-911.

chitectures provides a framework to describe widely differing implementations at a level of abstraction that eliminates many algorithm-related details, while preserving the important implementation characteristics. We expect that most future implementations in large RAMS will use the test-architecture-based approach, since it can easily adapt to changes in technology. ■

Acknowledgments

This work was supported by the University of Wisconsin Graduate Research Committee, an IBM graduate fellowship, and the National Science Foundation under contract MIP 8509194. We thank the referees for their comments and suggestions, which greatly enhanced the quality of presentation of this work.

References

1. A.J. van de Goor and C.A. Verruijt, "An Overview of Deterministic Functional RAM Chip Testing," *ACM Computing Surveys*, Vol. 22, No. 1, Mar. 1990, pp. 5-33.

2. M. Franklin, K.K. Saluja, and K. Kinoshita "Built-In Self-Test Algorithm for Row/Column Pattern Sensitive Faults in RAMs," *IEEE J. Solid-State Circuits*, Vol. 25, No. 2, Apr. 1990, pp. 514-524.

3. K. Kinoshita and K.K. Saluja, "Built-In Testing of Memory Using an On-Chip Compact Testing Scheme," *IEEE Trans. Computers*, Vol. 35, No. 10, Oct. 1986, pp. 862-870.

4. Y. Matsuda et al., "New Array Architecture for Parallel Testing in VLSI Memories," *Proc. Int'l Test Conf.*, Computer Society Press, Los Alamitos, Calif., Order No. 1962, 1989, pp. 322-326.

5. T. Sridhar "New Parallel Test Approach for Large Memories," *Proc. Int'l Test Conf.*, Computer Society Press, Los Alamitos, Calif., Order No. 641 (microfiche only), 1985, pp. 462-470.

6. M.M. Breuer and A.D. Friedman, *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Potomac, Md., 1976.

7. T. Ohsawa et al., "A 60-ns 4-Mbit CMOS DRAM with Built-In Self-Test Function," *IEEE J. Solid-State Circuits*, Vol. 22, No. 5, Oct. 1987, pp. 663-668.

8. P.H. Voss et al., "A 14-ns 256K × 1 CMOS SRAM with Multiple Test Modes," *IEEE J. Solid-State Circuits*, Vol. 24, No. 4, Aug. 1989, pp. 874-880.

9. F. Miyaji et al., "Multibit Test Trigger Circuit for Megabit SRAMs," *IEEE J. Solid-State Circuits*, Vol. 25, No. 1, Feb. 1990, pp. 68-71.

Manoj Franklin is a PhD student in computer science at the University of Wisconsin--Madison. He has been awarded an IBM graduate fellowship. Recently he was a summer intern at Cray Research. His research interests include high-performance computing, memory testing, built-in self-test, and design for testability. Before graduate studies, he was an engineer at BHEL, Bangalore, India.

Franklin received his BSc (engineering) in electronics and communications from the University of Kerala, Trivandrum, India, in 1984.

Kewal K. Saluja is an associate professor in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison, where he teaches logic design, computer architecture, microprocessor-based systems, VLSI design, and testing. Previously, he was at the University of Newcastle, Australia. He has also held visiting and consulting positions at the University of Southern California, University of Iowa, State University of New York, and Hiroshima University. His research interests include design for testability, fault-tolerant computing, VLSI design, and computer architecture.

Saluja received his BE in electrical engineering from the University of Roorkee, India, in 1967, and his MS and PhD from the University of Iowa in 1972 and 1973.

Saluja can be contacted at the Department of Electrical and Computer Engineering, University of Wisconsin, 1415 Johnson Dr., Madison, WI 53706.

NEW

IEEE EDUCATIONAL ACTIVITIES INTRODUCES

ICASSP/90

The following four video tutorials were recorded April 1990 in Albuquerque, New Mexico at ICASSP/90.

- The Structure of FFT and Convolution Algorithms, presented by James W. Cooley, IBM T.J. Watson Research Labs
- High-Resolution and Higher-Order Spectral Analysis, presented by Larry Marple, Chief Scientist, ORINCON Inc.
- Synthetic Aperture Radar: A Signal Processing Viewpoint, presented by David C. Munson, Jr., University of Illinois
- VLSI For Signal Processing, presented by Edward A. Lee, UC Berkeley and K. Wojtek Przytula, Hughes Research Laboratories

ICASSP/90 Package (All Four Programs) IEEE member price, $225.95, list $129.50. Plus shipping and handling.

Individual tutorials IEEE member price, $65.95, list $135.95. Plus shipping and handling. PAL video standard available upon request.

VLSI Design Principles and Practices

This Self-Study Course provides practical application of comprehensive information on circuit and logic design offering an opportunity to become a valuable interface for VLSI selection.

Program includes a study guide, answer book, applications workbook, diskette, and a textbook, VLSI Handbook by Joseph DiGiacomo, McGraw Hill, 1989.

VLSI is IEEE member price, $249, list $498. Plus shipping and handling.

For a full description of these programs and complete ordering information, call IEEE at 1-800-678-IEEE, FAX# 201-981-1686, or Telex: 0833233. For shipments to CA, DE, NJ, and NY add appropriate sales tax. Please call for appropriate overseas Air Mail Charges.

IEEE Educational Activities, 445 Hoes Lane
PO Box 1331, Piscataway, NJ 08855-1331

Chain Reactions in Networks

Udi Manber
University of Arizona

Theoretically, it's impossible to prevent the berserk loops of messages that can bring down an entire computer network. But a carefully applied set of rules will minimize these disasters.

Computer programs are not always correct and computer hardware is not always fault-free. A hardware failure can easily bring down a computer, and software errors can sometimes do that too. Most computer users are accustomed to periodic shutdowns; except for applications requiring a high level of reliability, infrequent shutdowns, though undesirable, are acceptable. Now, however, since most computers are connected by a myriad of communication networks, there is the danger that a design failure either in one computer or in one network protocol will propagate throughout a network, causing a breakdown of the entire network. This is clearly unacceptable.

We use the notion of chain reactions to characterize these breakdowns. A *chain reaction* is defined by *Webster's New World Dictionary* as a "self-sustaining series of chemical or nuclear reactions in which reaction products keep the process going." Except for restricting the reactions to chemical or nuclear reactions, this definition fits our subject perfectly.

The increase in network availability has brought numerous network applications that require complicated protocols and distributed algorithms. This added complexity substantially raises the potential for design failures. The main purpose of this article is to highlight some failures and discuss their prevention. Unfortunately, there are no easy ways to safeguard software systems. But the more we understand the weaknesses, the better equipped we will be to protect against them.

This article identifies a set of situations that are susceptible to network disasters. It presents several examples of actual breakdowns, which, although they occurred in different types of networks, have great similarities. It also gives several examples of potential breakdowns that can result from very subtle design failures. A general program that can prevent or even detect all potential chain reactions in an arbitrary protocol is impossible to design (see Theorem 1), but this article does suggest several steps to minimize the problem. Much of this discussion is part of the "folklore" among protocol designers. My purpose here is to unify some ad hoc observations and solutions. It is widely believed that distributed programs in general can have very subtle design failures that are hard to detect. This article supports that belief, but it also helps in understanding and preventing errors. This is especially important since network protocols are becoming more and more diverse, interleaved, and complicated.

A chain reaction in an Ethernet

The first example involves a local area network. The problem was caused by an error in a network protocol of a new operating system, Berkeley Unix 4.3. The error occurred when the operating system was installed on workstations (Microvaxs) connected by an Ethernet that also connected workstations still running the previous version (4.2). In describing details of the

0018-9162/90/1000-0057$01.00 © 1990 IEEE

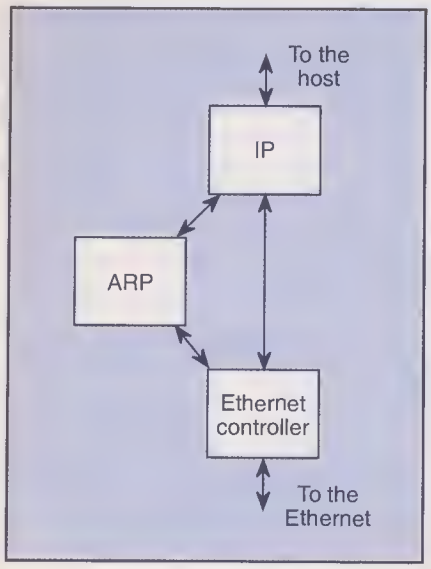


Figure 1. The network protocols involved in the Ethernet problem (IP=Internet Protocol; ARP = Address Resolution Protocol).

protocols used, the discussion is limited to the way the protocols were used in that particular case and to the parts relevant to the breakdown.

Background. An Ethernet is a broadcast medium in which all messages reach all destinations. Each host has an Ethernet controller, which is responsible for determining which messages are addressed to its host and thus should be picked from the network and forwarded to the host. The messages picked by the controller are either messages addressed to the host or broadcast messages. The protocol to which the controllers delivered the messages they picked was the Internet Protocol (IP), which serves as the internetwork layer in the Internet. IP is a complicated protocol, responsible for many tasks. In particular, IP was connected to another protocol, the Address Resolution Protocol (ARP), which is used to translate between high-level Internet addresses and low-level physical addresses. Each message had two addresses: an IP address and an Ethernet address. ARP was used as an interface between the two. Both IP and ARP are general-purpose protocols designed for large networks, but they can also be used (and often are) inside only one local area network.

Figure 1 illustrates the connections among the Ethernet controller, IP, and ARP. A correct message sent on the Ethernet has the right Ethernet address and the right IP address. It is picked by the appropriate controller using the Ethernet address and forwarded to IP. IP checks the IP address and then handles the message (for example, it assembles packages) and delivers it to the host through other protocol layers. IP also receives messages from the host and uses ARP to find the correct Ethernet address for them.

The breakdown. The chain reaction arose from an inconsistency between the conventions for specifying broadcast messages in the new operating system and the one it replaced. In Berkeley Unix 4.2 a broadcast is denoted by all 0's in the IP host field; in 4.3 it is denoted by all 1's. The network in question had both 4.2 and 4.3 machines. Consider the fate of a 4.3 broadcast message, which had a 4.3 IP broadcast address with all 1's in the host field. This message was picked up by the 4.2 Ethernet controllers (because its Ethernet broadcast address was correct), and forwarded to the 4.2 IP. The 4.2 IP did not recognize the IP address with all 1's.

Being a good citizen, IP tried to resolve the apparent error. Fortunately, IP had a mechanism for doing just that: It asked ARP to find the correct Ethernet address for this IP address so that the message could be forwarded to its intended destination. ARP, however, had no information about that particular address, so it created a request message to all the other ARPs, saying in essence, "Does anyone know how to translate this address?" Since no ARP could help, these messages were discarded.

So far the results of this error were not disastrous. Some broadcast messages were not delivered, and they caused several error messages and some unnecessary load on the protocols. The effects of the error were compounded, however, by the presence of programs that generated broadcast messages frequently (such as *rwho*). Networks with both 4.2 and 4.3 machines did slow down (sometimes significantly), but they didn't crash. It is the following attempt to correct the problem that makes this example more interesting.

It was clear to the people monitoring the message traffic that the problem was caused by messages to a certain IP address (with all 1's), which for some unknown reason generated numerous ARP requests. At a certain site the following quite natural solution was attempted: If ARPs cannot figure out that the message is a broadcast message, let's tell them. So an entry was made (by hand) in one ARP's table, translating that IP address to the Ethernet broadcast address. The idea was that ARP requests would reach this one ARP that could resolve them, and the original messages would be made into the broadcast messages they were supposed to be.

The result was quite different. Consider one 4.2 IP that does not recognize the 4.3 IP message as a broadcast message and asks its ARP to help. ARP finds out that the message is indeed a broadcast message, so IP forwards the message with an Ethernet broadcast address. But this forwarded message still causes the same problem. The 4.2 IPs do not recognize its IP address, which remains the same; they use their ARP to resolve the address; the ARPs resolve the address; and more broadcast messages are generated. If there are k 4.2 machines, then each original broadcast message will be regenerated k times, each of those k broadcast messages will be regenerated k times, and so on.

The result was that the network was swamped with these broadcast messages, and several machines crashed very quickly. Not only that, but the only way to correct the problem was to physically disconnect the machines and restart everything (without the extra entry in the ARP table, of course).

Infinite loops of mail messages

Background. When a user moves from one place to another, or even from one computer to another at the same site, the mail should be forwarded appropriately. This can be done by using forwarding commands; for example, in Unix one can specify the forwarding address in a file named .forward. Before the mail is delivered, the .forward file is consulted and, if it is not empty, the mail is automatically forwarded to the address in the file. Suppose that a careless user forms a cycle of forwarding pointers; for example, machine A has a .forward file pointed to machine B and machine B has a .forward file pointed to machine A. (Unfortunately, this is not unusual; a user may decide to receive the mail at site B, and the easiest way to do so is by adding a .forward file at site A to point to B. That user may not be aware that site B already has a .forward to site A.)

Let's call a user whose mail path contains a cycle a *cycled user*. If no action is taken, a mail message sent to a cycled user will

bounce from one site to another in an infinite loop. In practice, mail messages are *aged*. The number of hops that a message can take has a limit (for example, 30). When the limit is reached, the attempt to deliver the message is ended, and an error message is generated. (See a later section for more detail on aging messages.)

The breakdown. Although forwarding cycles are not uncommon, they do not cause any problems except for the cycled user, who, of course, cannot get any mail. Consider, however, the following scenario, which actually happened. John, who was (unknowingly) a cycled user, noticed that mail was not coming through. His knee-jerk reaction was to send a test mail message to himself to see what would happen. Well, let's see what happened: The mail bounced in the cycle until the limit was reached; then, an error message was generated. An error message is addressed to the sender, who, in this case, was also the intended recipient. Of course, the same fate awaited the error message; it bounced in the cycle until the limit was reached, and another error message was generated, and so on. And, as if that was not bad enough, the messages were in fact getting longer. An error message contains a header that identifies it as an error message and identifies its cause, and it also contains the original message. The growth was linear and not exponential, but it was sufficient to crash the system.

The two machines involved in the cycle were connected by a fast local area network. The messages bounced back and forth rapidly and grew rapidly, until they filled the available buffers and crashed the machines. If the two machines had been connected by a slow network, the loop would have consisted of messages going back and forth every few minutes (or hours). Such a loop would have degraded the performance of the network slowly but could have remained undetected for a long time.

Another variation of an infinite message loop happened at the University of Arizona in May 1989. As a result of an erroneous setup of an initialization file (the details are immaterial to this discussion), a workstation named Corona, which was connected to a file server named Baskerville, "thought" that it was Baskerville itself. When Corona was rebooted, it tried to establish connection with, among others, Baskerville. The connection was unsuccessful because Baskerville, being smart, did not want to talk to itself (the message Baskerville received from Corona was something of the form "Hi, I'm Baskerville."). An error message was generated about the failed connection. The error message was addressed automatically to a mailing list (maintained on a computer called Megaron) that included the staff responsible for the mail. Unfortunately, Corona was also the personal machine of a staff member who received her mail there and who was on the mailing list. The error message thus could not be delivered, and again an error message was generated, addressed to — you guessed it — the staff.

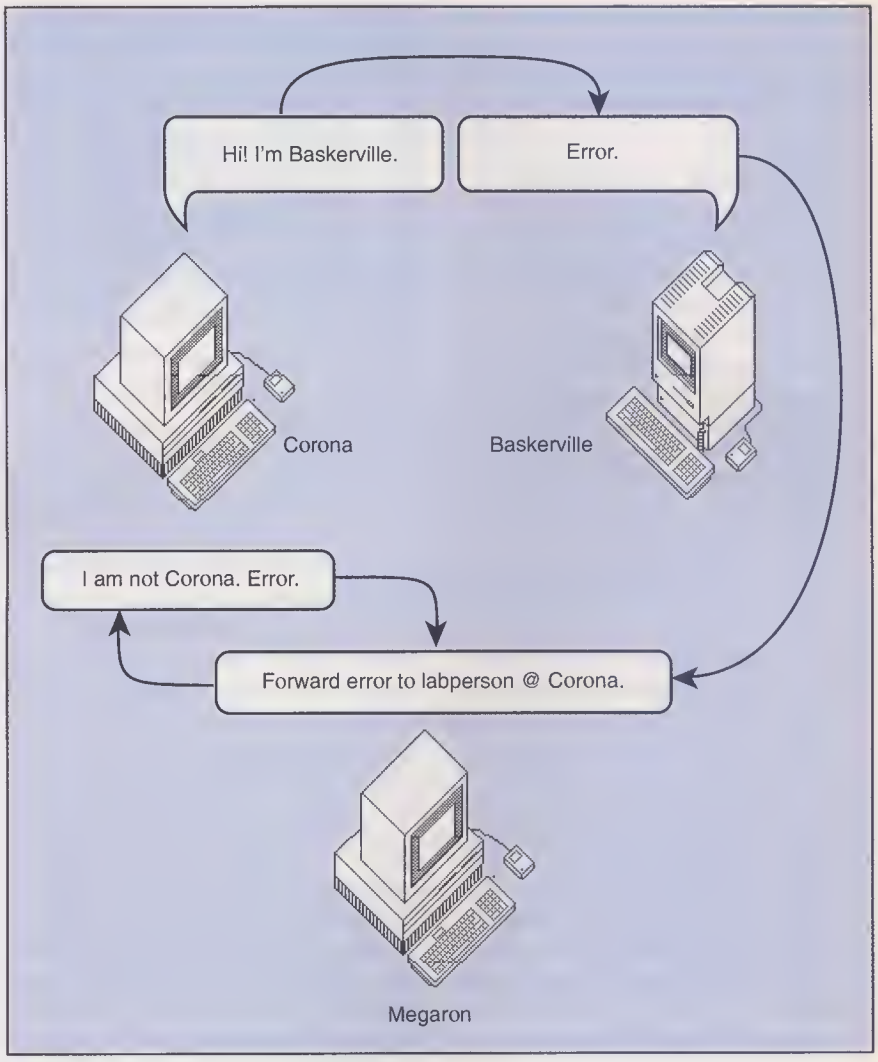


Figure 2. An infinite loop of messages.

Figure 2 shows this chain of events. Notice that, because several people were involved, the resulting loop was more serious than the one previously described. Each error message was sent to the whole mailing list. The result was a crash of several machines—the ones on which the staff members received their mail—because their disks got full. Again, the error messages got bigger and bigger, thus speeding the process.

An even more serious loop might have been formed if two staff members had their mail delivered to Corona or if another machine belonging to a staff member had been rebooted at the same time (the setup error was identical in most machines). In those cases every error message would have been sent to both members, which would have generated two error messages, and so on. This example illustrates the danger of multiple forwarding or any use of large mailing lists, especially ones that are supposed to receive error messages. It is more prudent to deposit the errors in a file and have whoever is responsible look at that file occasionally, rather than to use

the normal mail program. Huge mailing lists with users all over the world are quite common today. Failures involving such mailing lists can be disastrous.

Additional examples

Four more examples of real or potential chain reactions further illustrate how fragile network protocols can be.

The most famous example of a chain reaction occurred in the Arpanet in 1980. A hardware failure that caused bit dropping in one host caused a loop of routing update messages that brought down the whole network. This breakdown is discussed in detail by Rosen.[1]

From our earlier discussion of local area networks, recall that a message is always broadcast but that only the intended destination actually reads the message (if everything works properly). Now suppose that we want to connect several local area networks, using bridges, into one network, something that is becoming very common. Suppose that we want to keep the idea of broadcasting messages throughout the combined network. If the combined network does not contain any cycle, a flooding algorithm works very well. That is, every bridge that connects several networks forwards messages to all the networks connected to it except the one from which the message came. If, on the other hand, the interconnection contains cycles, then we can use a spanning tree (a connected subgraph containing all the nodes but no cycles) for the broadcasting (see Figure 3).

Perlman gives a protocol for finding a spanning tree in such an environment.[2] Perlman and Varghese present an example of a seemingly straightforward improvement to Perlman's protocol and show that this "improved" protocol can cause an exponential number of messages.[3] We cannot describe the protocol in any detail here; we only highlight its main deficiency. The main step of the protocol is for each bridge to send a Loop-Detect message that is flooded to all the nodes. If the bridge is contained in a cycle, then the Loop-Detect message returns to it, and it can take the appropriate actions. The protocol is correct in the sense that all Loop-Detect messages will eventually be absorbed and no infinite loops can occur. The problem is that for some networks, such as the network in Figure 3, such flooding generates an exponential number of messages. There is a real danger that so many messages will be generated before any of them is absorbed by the algorithm that the network will be swamped and become unusable. The moral of this example is that guaranteeing that no infinite loops occur is not sufficient; we must ensure that not too many messages are generated.

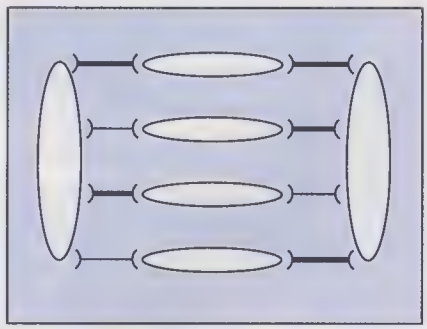


Figure 3. Connecting several local area networks; the bold lines correspond to the spanning tree.

Another example is a protocol for queries in a large network. A node can send a query message to other nodes, and they, in turn, either answer the query (if they know the answer) or forward the query. Suppose that each query has a unique identifier and that each node remembers the queries it has seen. The queries are forwarded, possibly in a floodlike manner, throughout the network, but the identifiers are used in an absorption algorithm so that a node never forwards the same query twice. Suppose now that someone is attempting to improve the efficiency of the protocol in the following way: A query may contain several parts. If a node can answer part of the query, it does not forward the whole query, only the parts that still need to be answered. Suppose that in the implementation the identifiers of the partial queries are new (which is natural since the partial queries are different messages). This cannot cause infinite loops because a query with k parts can be changed only $k - 1$ times (it is becoming smaller all the time). However, even though the number of changes to any query is $k - 1$ (and k is probably small), the number of possible partial queries is 2^k (all subsets of at most k parts). So, in the worst case, 2^k messages can be generated from one query and broadcast to the whole network, an unacceptable situation.

The final example is very different from all the rest. It is the Internet "worm" that attacked thousands of computers across the United States in November 1988 (see Spafford[4]). In this case the "protocol" had no useful purpose, and the chain reaction may have been initiated intentionally, but there is some reason to believe that it was not. It could have been caused by sheer carelessness, which is what this article is meant to prevent. The two characteristics of the worm that are of interest to us are that it attempted to spread to as many computers as possible, and that whenever it established itself, it regenerated periodically, making it harder for someone to destroy it. This is clearly a recipe for a chain reaction. The main point is that there was no mechanism to limit the number of times a host could generate more copies of the worm. Such a mechanism is essential in any distributed program.

Detecting and preventing chain reactions

The examples shown so far have quite a bit in common. This section studies the similarities, identifies the "culprits," and suggests ways to prevent such occurrences. It shows that it is theoretically impossible to prevent *all* possible failures but presents some guidelines to minimize them.

The common factor in all the examples was an infinite or a very large exponential loop. The loop consisted of messages—control messages, normal messages, or broadcast messages—that generated other messages. The loops in the examples were all results of subtle design failures. The first question is whether it is possible to detect such loops. Can we somehow analyze the protocols and find situations that can potentially cause loops? Unfortunately, the answer to this question is no.

Impossibility of chain reaction detection. First, let's define a very simple model of distributed computation. There are N machines, each executing a local program (protocol). The programs are sequential with two additional message-passing primitives called Send and Receive. A Send instruction moves some data to another machine (specified in the instruction) and deposits it in some buffer in the other machine, and a Receive instruction fetches messages and acts on them. For simplicity, I ignore many issues, such as synchronization, interrupts, connectivity (assume all machines are fully connected to all others), and size of the buffers.

The question addressed is whether it is possible to determine, given the programs

and their inputs, whether the number of messages will be finite. The answer to this question is negative because this problem is equivalent to the halting problem. The *halting problem* is to determine, given a (sequential) program and its input, whether or not the program terminates. This problem is *undecidable*, meaning no program can solve it in finite time (see, for example, Hopcroft and Ullman[5]). The proof that it is impossible to determine whether the number of messages is finite follows directly from the undecidability of the halting problem:

Theorem 1: Given a distributed program as defined above, it is impossible to determine whether the number of messages generated by the program is finite.

Proof: Suppose that we had a program *P* that could determine, under the model presented above, whether the number of messages is finite. We could then use *P* to solve the halting problem. Given a sequential program *S* for which we want to solve the halting problem, we insert after every instruction of *S* a Send instruction (to an arbitrary machine). Clearly, the number of messages emanating from *S* is finite if and only if *S* terminates.

In practice, the fact that the halting problem cannot be solved does not severely handicap programmers (at least not directly). We can usually estimate how long a program should take, and if the program runs too long, we simply stop (kill) it. But a major difference between sequential computation and distributed computation is that it is not easy to stop a distributed program that goes berserk. We cannot simply hit a "kill" button and stop the program because many different computers may be involved. Furthermore, in many cases the only way to access these computers from the place that the problem originated is through the network. But the network is unusable as a result of the problem. That is why preventing chain reactions is so important.

Similarities of the examples. In the Ethernet example the problem was that broadcast messages were not identified as such and were forwarded. A loop containing broadcast messages is obviously much more "explosive." But the heart of the problem was the inability of the forwarding protocol to realize that it was attempting to forward the same message again and again.

In the next example the loop of mail messages was caused by error messages generated from other error messages. Again, the error-handling mechanism was not smart enough to detect that the same error was being "reported" repeatedly. Notice that in this case it was not the same message that was forwarded, because each time a new error message was generated.

The inherent problem that caused the breakdowns is automatic generation of messages, or AGM.

The problem in the example of loop detection in a collection of local area networks was not the inability to detect cycles. The algorithm was designed to detect all cycles eventually. The problem was rather the exponential proliferation of messages before the cycles were detected and broken. The example shows that in some cases even temporary cycles can cause a breakdown.

AGM. The inherent problem that caused the breakdowns in these examples is what I call *automatic generation of messages*, or AGM. I include in the definition of AGM any message generated by a program triggered by a message from another computer. (Variations of this definition are discussed later.) Thus, forwarding a message is an AGM. Using AGM can be highly risky, as the examples show, but unfortunately, it is necessary. For example, the frequent dissemination of routing information cannot be done without AGM, error messages are always generated automatically, and a .forward file is a very convenient feature requiring AGM. However, AGM makes network protocols very vulnerable. Even simple AGM can lead to chain reactions. Here is another example:

A vacation program is a popular program that reads one's mail and automatically replies with a message saying something like "I am on vacation; I will reply when I return." It is a very easy program to write, but, since it is an AGM program, it should be written with care (see Allman[6]). For example, if the message generated by the vacation program results in an error message, then this error message will receive the usual vacation reply, and an infinite loop will result. A good vacation program should remember all addresses to which it replied, and it should not reply to those again (at least not within a specified period). Even then, if the program fails to recognize the address of the original message (conventions for addresses can vary), then infinite loops are possible.

AGM control methods. Methods for dealing with AGM are often ad hoc and incomplete. The most common method of safeguarding against infinite streams of messages is *aging*. Each message has an age field, which is incremented periodically as the message is forwarded, until the message becomes too old. There are several variations for deciding when to increment the age field. The aging may depend on the number of hops the message has taken. If the message goes through too many hops, it is discarded. This is the method that was supposed to prevent the loops of mail messages described earlier. Another possibility is to use the actual time that the message was generated and discard the message at a later time, when it becomes too old in the regular sense of the word. This method requires clock synchronization and thus may be suitable only for coarse-grained timing (in minutes, for example).

In many cases, however, aging is not a sufficient safeguard. The main reason that the aging process can be circumvented is that AGM not only forwards messages but often generates totally new messages. A new message is almost by definition a young message. This is exactly what caused the infinite loops of messages in our earlier example. The aging process detected the original loop, but then an error message—newborn as far as the aging process could determine—was generated, causing another loop. One can argue that error messages should be created with the age of the message they report on, but that is not so easy. For example, an error message reporting that a message has reached the bound on the number of hops cannot start with the already maximal bound on the number of hops; it will not get anywhere. Maybe we should set the number of hops of that error message to be the maximal number minus *k*, where *k* is a small number (so that we allow *k* hops for this error message). But in that case the same infinite loop of messages will occur.

Relying on *time-outs* is another common approach. If a message does not arrive at its destination within a specified time, it is discarded (in reliable communication a copy of the message is kept, and if an acknowledgment is not received, the message is attempted again). Although this approach prevents old messages from existing on the

network, it does not prevent the creation of young messages. Another major drawback is that chain reactions that exhibit exponential growth can flood the network faster than the time-out, as was described earlier.

Another approach for preventing infinite AGM is *absorption*. Messages are allowed to multiply, but they are absorbed under some rules. A rule commonly used in flooding is to attach an identification number to each message and to remember all message identifiers. If a message arrives twice, it is discarded. The main problems with this approach are how to generate the identifiers and how long to remember them.

A relatively safe way to minimize chain reactions. By combining the methods just discussed, we can obtain an algorithm that prevents chain reactions under most circumstances. Several similar algorithms have been suggested (see, for example, Perlman[7]). This algorithm is less specific than others in that it addresses all AGMs, not only those that occur in the protocols under discussion. Because of its generality, it may be less efficient than algorithms for specific protocols, which can be tuned up. But addressing all AGMs is important enough to warrant the extra cost. At the least, this algorithm can be used as a guideline toward the goal of preventing chain reactions.

As noted, AGM is the root cause of chain reactions but is essential to network protocols. We cannot (and do not want to) restrict the use of AGM. Also, we cannot, in general, distinguish between "bad" AGMs (that is, those that cause chain reactions) and "good" AGMs. I believe AGMs should always be treated as potential chain reactions. We have seen examples of chain reactions resulting from simple error messages and from simple forwarding of "lost" messages (which turned out to be broadcast messages). Any time a protocol includes generating a message as a reply to another message, the protocol should be tested very carefully. The difficulty is that protocols are very complicated and highly interrelated. It is sometimes impossible to foresee all possibilities.

If we consider any AGM a possible risk, it is clear that we need a general absorption algorithm, which will be uniformly applied to all AGMs. It is possible, of course, to adapt the absorption algorithm to the specific protocol, and the algorithm may then be more efficient, but a general algorithm suitable for all protocols is also desirable. The following four rules constitute such an algorithm. Implementation issues, extensions, exceptions, and relaxations of some of these rules are discussed below.

We need a general absorption algorithm that will be uniformly applied to all AGMs.

(1) A unique identifier should be associated with every message in the network. In some cases there is already enough information in the message header to serve as the identifier (for example, host name, sender, and date).

(2) Whenever a message M_2 is being generated as a result of another message M_1 (in other words, an AGM occurs), then the identifier of M_2 should be the same as that of M_1. (M_2 may include other identifiers as well, but for our purposes, we consider its identifier as that of M_1.) This rule should hold for every AGM, regardless of how simple the protocol is or how infrequently it is used.

(3) The identifiers of all messages for which an AGM has occurred in a given host should be kept by the host.

(4) When a message with an identifier that exists in the host's table returns to the same host, no further AGM should occur.

These rules, if implemented correctly, are sufficient to prevent chain reactions:

Theorem 2: Suppose that the rules above are maintained by all protocols used in the network. Let H be a bound on the total number of messages created by hand (that is, by a human) in the whole network and L be the number of links in the network. Then, the total number of messages that can be delivered in the network is at most $2LH$.

Proof: Rule 2 guarantees that all messages created by a program have the same identifiers as messages created by hand. Therefore, in a sense, programs do not create any new messages (where "new" indicates new identifiers). Every message M that is created by hand can be forwarded at most twice along any link (once in each direction). The rules guarantee that the same node will not forward the same message more than once. Hence, the total number of messages in the network cannot be more than $2LH$.

These rules may not eliminate chain reactions completely, because there is no guarantee that they will be implemented correctly or that the hardware will be immune to failures. (If the hardware is susceptible to arbitrary failures, no algorithm is safe.) Verifying the correctness of protocols is an extremely difficult problem, which is not discussed here.

The rules may seem inefficient and limited for several reasons. First, requiring that all messages that ever passed through the network be remembered is not reasonable. This requires too much memory and also requires the identifiers to be quite large. Fortunately, we can relax the rule to state that only the identifiers of messages arriving in the last T time units should be kept (for example, the last hour). The value of T will depend on the speed and size of the network. The relaxed rule will not prevent chain reactions because infinite loops will still be possible, but it will slow them down considerably. If T is an hour, then each loop can be repeated once an hour at most, which minimizes the problem. Remembering only recent messages does not require too much memory, nor does checking for duplication require too much of a load. (It is not generally sufficient to remember only the last k messages, for some small value of k. If the chain reaction involves more than k messages to the same host, it may not be prevented.)

The second problem is the generation of identifiers. A message typically passes through several layers of protocols, each of which is quite independent, and several of which may be involved in AGMs. If an identifier is generated at a low-level protocol, the higher-level protocols may not be able to recognize the identifier because it is part of the message to which they have no access. To prevent chain reactions, identifiers should be generated as early as possible in message creation (for example, by the application) and be written in a place accessible to all the layers involved in AGMs. (It is possible to have identifiers for each such layer, although it is not necessary.) This requirement may violate the modularity and independence of the layers, but it is important enough to warrant consideration.

The third problem with the rules is also caused by the multitude and independence of the protocol layers through which messages pass. A message that arrives at a certain machine may be involved in multiple AGMs by different layers in the same machine. The definition of an AGM applies only to messages from another machine, but the origin of the message is not always easy to determine. If only one table

is maintained per host (as suggested in rules 3 and 4), then the second AGM in the same machine will be stopped by rule 4. We can avoid this problem either by maintaining a table for each layer or by not counting as AGM messages that are moved in the same machine.

The fourth problem is that error messages are severely restricted by rule 4. If an error is detected by a node that has already participated in the forwarding of the message, that node cannot generate an error message. This rule can be relaxed in the following way: We can define two (or more) types of messages: normal messages and error messages. An error message created as a result of a normal message keeps the same identifier, but it is labeled an error message. The rules are modified to include the types. In particular, rule 4 now states that a node does not forward a message with the same identifier *and the same type* more than once. Notice that this rule forbids creating error messages as a result of error messages.

The fifth problem is the time and space required to implement the rules. Some protocols, such as routing protocols, rely on very efficient implementation and they may not afford the extra work. We can relax some of the rules, but that will increase the possibility of chain reactions. For example, when a message from A to B passes through intermediate nodes, these nodes may not need to record this fact as rule 3 requires. We can think of this message as going directly from A to B.

The governing principle behind these rules is that they should be applied to *all* protocols. Some protocols already apply some (or all) of these rules. For example, the Internet Control Message Protocol (ICMP), a protocol responsible mainly for error messages, does not allow an error message to be generated from an error message (see Comer[8]). The vacation program that is part of the Berkeley Unix distribution keeps track of the addresses it has responded to, and its default is to send no more than one response per address per week. Protocols can be interrelated in very complex ways, however, so it is not sufficient to protect a protocol as if it operates in a vacuum.

The rules pose restrictions on protocol design. In some cases a node may need to forward the same message, or its relatives, several times. In most of these cases one can set a bound on the number of forwardings and apply the trick of having several (but a small number) of message types. Identifying these cases and assigning the types is important even if the suggested rules are not followed. For example, consider the query protocol discussed in "Additional examples." Suppose that a node tries to send an answer to a query back to the originator of the query. If that answer message passes through a node that participated in the forwarding of the query, then, according to our rules, the answer should not be forwarded (the answer is an AGM of the query and thus has the same identifier). The solution to this problem is clear. There should be two types of messages, a query and an answer.

We can relax rule 4 slightly such that messages that have already passed through the host are not dropped but rather buffered for a while (an hour, for example). This will remove some of the restrictions (such as the restriction against error messages in response to error messages), and it will limit the speed of possible chain reactions. It is essential, however, to check before buffering that a message with the same identifier is not already buffered. This variation of rule 4 can lead to chain reactions, so it should be implemented with great care and only as a last resort.

Chain reactions are unusual, and most of the examples presented here were caused by improbable sets of events. Nevertheless, because the cost of recovering from a network breakdown can be high, we should do whatever we can to prevent such breakdowns. We cannot foresee all possible interactions between network protocols, and we cannot hope that the probabilities will always work in our favor. The simple set of rules given in this article will prevent most chain reactions. I believe that these rules, or variations of them, should be incorporated in all network protocols. ■

Acknowledgments

The work on which this article is based was supported in part by an NSF Presidential Young Investigator Award (grant DCR-8451397), with matching funds from AT&T.

I thank Dave Oran and Larry Peterson for many helpful conversations, suggestions, and observations about chain reactions in networks. I also thank Greg Andrews and the anonymous referees for comments that improved the article.

References

1. E.C. Rosen, "Vulnerabilities of Network Control Protocols: An Example," *Computer Comm. Review*, July 1981, pp. 10-16.

2. R. Perlman, "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN," *Proc. Ninth Data Comm. Symp.*, Vancouver, Canada, 1985, pp. 44-53.

3. R. Perlman and G. Varghese, "Pitfalls in the Design of Distributed Routing Algorithms," *Proc. ACM SIGComm Symp. Comm., Architectures, and Protocols*, ACM, New York, 1988, pp. 43-54.

4. E.H. Spafford, "The Internet Worm: Crisis and Aftermath," *Comm. ACM*, Vol. 32, June 1989, pp. 678-688.

5. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass., 1979.

6. E. Allman, "Sendmail—An Internetwork Mail Router," in *Unix System Manager's Manual*, 4.3 Berkeley Software Distribution, Univ. of California, Berkeley, Apr. 1986.

7. R. Perlman, "Fault-Tolerant Broadcast of Routing Information," *Computer Networks*, Vol. 7, 1983, pp. 395-405.

8. D. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Prentice Hall, Englewood Cliffs, N.J., 1988, pp. 90-91.

Udi Manber is a professor of computer science at the University of Arizona, where he has worked since 1987. His research interests include design of algorithms, distributed and parallel computing, and computer networks. He is the author of *Introduction to Algorithms—A Creative Approach* (Addison-Wesley, 1989).

Manber received his PhD degree in computer science from the University of Washington in 1982. He received a Presidential Young Investigator Award in 1985, the Best Paper Award of the Seventh International Conference on Distributed Computing Systems, 1987, and a Distinguished Teaching Award of the Faculty of Science at the University of Arizona, 1990.

The author's address is Dept. of Computer Science, University of Arizona, Tucson, AZ 85721; his e-mail address is udi@cs.arizona.edu.

NEED TO KEEP UP TO DATE WITH COMPUTER SCIENCE?

Computer Science is a rapidly-developing field of research. You need high-quality scientific information about the latest developments and research results in order to keep up to date.

North-Holland offers you an extensive high-level publication programme with journals and books covering a wide range of subject areas, such as: Information Systems, Computer Communications, Distributed Systems & Telecommunications, Theoretical Computer Science & Database Theory, Artificial Intelligence, Expert Systems, Knowledge Engineering, Computer Architecture, Parallel and Supercomputing, Computer Performance, Simulation and Modelling, Industrial Automation, Computer Graphics, Pattern Recognition, Computers in Education, Computers & Security.

North-Holland's Computer Science Catalogue 1990 contains full details of our latest books and our range of journals. Write for your personal copy of the catalogue by completing the coupon!

NORTH-HOLLAND PUBLICATIONS

North-Holland is an imprint of Elsevier Science Publishers BV, one of the world's largest scientific publishers.

COUPON *COMP/0*

Send this coupon - or a photocopy - to:

North-Holland
(An Imprint of Elsevier Science Publishers BV)
Attn.: Petra van der Meer
P.O. Box 103
1000 AC Amsterdam
The Netherlands

Customers in the USA/Canada:

Elsevier Science Publishing Co. Inc.
P.O. Box 882
Madison Square Station
New York, NY 10159, USA

☐ Yes, please send me my personal copy of the **Computer Science Catalogue 1990**

Name _____
Address _____
Country _____

Reader Service Number 6

SPECIAL REPORT

The 1988-89 Taulbee Survey Report

David Gries and Dorothy Marsh
Cornell University

Significant findings emerge from the 1988-89 Taulbee report, based on a December 1989 survey of the Forsythe list of computing departments.* The survey focuses on the production and employment of PhDs who graduated in 1988-89** and the faculty of PhD-granting computing departments during the 1989-90 academic year. All 129 computer science departments (117 US and 12 Canadian) participated, including 29 of 32 departments offering the PhD in computer engineering.*** Throughout this report, CE statistics are reported separately so that comparisons with previous years can be made for CS. Nevertheless, we intend to merge all statistics for CS and CE within the next few years.

The highlights of the survey are as follows:

• The 129 CS departments produced 625 PhDs, an increase of 8 percent over the previous year. Of the 625 PhDs, 336 were American, 35 were Canadian, and 248 were foreign (meaning neither American nor Canadian). Nearly half, 309, stayed in academia, 181 went into industry, 24 entered the governmental sector, 56 went overseas, seven were self-employed, and nine were unemployed. The employment status of 39 of the PhDs was not known.

• 1,215 students passed their PhD qualifying exam in CS departments, an increase of 9 percent over 1987-88.

• No blacks, six Hispanics, and 87 women received PhDs.

The Computing Research Board's survey shows an 8 percent increase in PhDs granted. Of the 625 new PhDs, none are black, six are Hispanic, and 87 are women.

• The 129 CS departments have 2,550 faculty members — an increase of 123, or nearly one per department. There are 938 assistant, 718 associate, and 894 full professors.

• The 129 CS departments reported hiring 204 faculty and losing 161 (to retirement, death, other universities, graduate school, and nonacademic positions).

• Six assistant professors in the 129 departments are black, 20 are Hispanic, and 92 are female. Two associate professors are black, six are Hispanic, and 66 are female. Four full professors are black, seven Hispanic, and 30 female.

A number of additional interesting observations can be made. For instance, although the growth in PhD production to 625 didn't reach the expected 650-700, an increase of nearly 50 PhDs is substantial. This growth made it easier for departments trying to hire and more difficult for the new PhDs in their job searching.

There is still a large market. However, the new PhDs cannot all expect to be placed in the more mature departments, and more will take positions in the newer departments and the non-PhD-granting departments.

The title of the survey honors Orrin E. Taulbee of the University of Pittsburgh, who conducted the 1970-1984 studies for the Computer Science Board. The CSB is now called the Computing Research Board and integrates computer engineering with computer science.

* The Forsythe list is maintained by the Computing Research Board and includes all departments in the US and Canada that grant a PhD in computer science and computer engineering. This is the third year that CE departments have been included in the survey.

** 156 departments reported on an academic-year basis or did not specify; two reported on a 1989 calendar-year basis.

*** The departments of computer engineering at the University of Illinois at Champaign-Urbana, Johns Hopkins University, and New Mexico State declined participation in the survey.

0018-9162/90/1000-0065$01.00 © 1990 IEEE

Table 1. Department titles.

No.	Title
89	Computer science(s)
21	Electrical and computer engineering
11	Computer and information science(s)
8	Computer science and engineering
7	Electrical engineering and computer science
4	Electrical engineering
2	Computer engineering
2	Computing science
2	Information and computer science
1	Advanced computer studies
1	Applied sciences
1	Computational science
1	Computer engineering and science
1	Computer science and electrical engineering
1	Computer science and operations research
1	Electrical, computer, and systems engineering
1	Electrical engineering and computer engineering
1	Mathematical and computer sciences
1	Mathematical sciences

(Note: "Center," "division," "program," and "school"

Growth of PhD production seems to have slowed enough so that it does not appear that over-production will be a concern in the near future. However, there won't be enough retirements to offset new PhD production for some 10 years to come. (In the 158 departments, 22 faculty members died or retired last year.) We believe that many of the new PhDs would benefit from a year or two in a postdoctoral program; perhaps this would be the ideal time for the National Science Foundation to institute such a program in computer science and engineering.

The percentage of CS PhDs given to foreign students remained about the same at 40 percent. In CE, the percentage was much higher, 64 percent.

The field continues to be far too young — a problem that time alone is slowly solving. CS continues to have more assistant professors than full professors — a situation that puts an added burden on the older people. Nonetheless, there was substantial growth this year in the number of associate professors (as assistant professors were promoted). Still, the ratio of assistant professors to full professors in CS has not changed appreciably in four years.

As far as we know, no other field has this problem, as we have mentioned in previous Taulbee reports. In fact, most scientific fields are 80- to 90-percent tenured in many universities. In CS, this problem is more severe at the newer and lower ranked departments. In fact, the top 24 departments now have 223 assistant, 176 associate, and 290 full professors. The CE departments have far more full professors than assistant professors, mainly because many are older EE departments offering CE degrees.

As indicated above, blacks and Hispanics simply are not entering computer science and engineering. The time has arrived that something be done to rectify this situation. We hope the CRB will take the lead in introducing programs to encourage more participation from these minorities.

There was slight growth in the percentage of female PhDs in CS, to 14 percent from 10 percent. Still, there are too few women in the field, and our record of retention of women in the faculty is abysmal. There are 30 female full professors in the 158 CS and CE PhD-granting departments. As with blacks and Hispanics, we hope the CRB will help introduce programs to encourage more women to enter computing and to remain in academia. There are indications that the NSF is interested in this problem as well.

Methodological comments. Questionnaires were sent to 129 CS PhD-granting departments and 32 CE PhD-granting departments in October 1989. (Table 1 lists the department titles.)

The figures in this report are complete for CS. The accuracy of the report depends, of course, on the accuracy with which the questionnaires were filled out by representatives of the individual departments. Joint EE-CS and EE-CE departments had a particularly difficult time completing the questionnaires, since they went to the extra effort of extracting the CS or CE information for their departments.

As with most surveys, part of the data was not filled in or was obviously incorrectly entered. We took the liberty to adjust some figures and estimate a few others. For example, in a few cases, with 156 or 157 out of 158 departments reporting a figure in a field, we estimated that field for the others. Our goal was to make this report consistent, clear, and simple, without modifying the overall results in any way.

In some instances, we analyzed the data for the higher ranked departments as compared to the lower and unranked ones, using the 1980 survey done under the auspices of the National Research Council[1] for ranking. (We also included the two largest Canadian universities within the top 20.) The NRC survey[1] is now 10 years old, and many changes have occurred in CS since 1980 (such as the emergence of more than 60 PhD-granting CS departments). Nevertheless, this breakdown still provides some useful comparisons.

To draw meaningful conclusions regarding growth of the field (using older surveys), from time to time within this report we compare figures for the CS departments only, keeping figures for CE separate. As stated previously, we will combine CS with CE in a few years.

Throughout the report, figures for 1970-84 are taken from O.E. Taulbee,[2] for 1984-86 from Gries,[3,4] and for 1986-88 from Gries and Marsh.[5,6] It should be further noted that the figures for 1970-84 may not be accurate because not all departments completed questionnaires during those years.

Data on students

PhD production and its growth. The field of CS produced 625 PhDs in 1988-89, an increase of 48 (8 percent) over 1987-88 and an increase of 395 (172 percent) over 1980. Table 2 provides figures on PhD production for CS and CE, as well as for qualifying-exam passage and sizes of incoming classes. In the Table 2 column headed "No. of Depts.," the first number represents the departments reporting and the second number represents the total of known PhD-granting departments.

As mentioned before, CS PhD production increased 8 percent this year, compared to 24 percent last year. Future growth is expected, but this will probably remain under 10 percent per year. The 129 CS departments project a 38 percent increase next year, but they have always been overly optimistic when forecasting PhD production. Often, a graduate student appears on a department's list of expected PhDs for three years in a row.

In previous years, estimates of the annual need for new PhDs in CS ranged from 600 to more than 1,000. This year, for the first time, we have surpassed the minimum estimate of 600. It now becomes even more

Table 2. PhD production and its growth.

	Yr.	No. of Depts.	New PhDs	Ave. Per Dept.	Qualifying Exam Passage	Ave. Per Dept.	New PhD Students	Ave. Per Dept.
CS	1980-81	—	230	—	—	—	—	—
CS	1984-85	103 (109)	326	3.2	755	8.2	1,177	12
CS	1985-86	117 (118)	412	3.5	858	7.3	1,170	10
CS	1986-87	123 (123)	466	3.8	1,008	8.2	1,430	12
CS	1987-88	127 (127)	577	4.5	1,113	8.8	1,497	12
CS	1988-89	129 (129)	625	4.8	1,215	9.4	1,632	13
CS-CE	1986-87	145 (156)	559	3.9	1,168	8.1	1,621	11
CS-CE	1987-88	157 (161)	744	4.7	1,399	8.9	1,801	11
CS-CE	1988-89	158 (161)	807	5.1	1,441	9.1	1,993	13

Table 3. Sex and minority status of the PhDs.

	CS			CE			CS-CE		
PhD Minority Status	Male	Female	Total	Male	Female	Total	Male	Female	Total
White, not Hisp. origin	330	62	392	90	14	104	420	76	496
Black, not Hisp. origin	0	0	0	0	0	0	0	0	0
Hispanic	4	2	6	6	0	6	10	2	12
Other	204	23	227	66	6	72	270	29	299
Total	538	87	625	162	20	182	700	107	807

important that we monitor PhD growth as well as demand.

The following explains that real future growth will come mainly from the newer and smaller two thirds of the departments and not the more established one third.

In 1988-89, an average of 5.1 CS-CE PhDs were produced per department (see Table 2), with 21 departments producing zero, 19 producing one, 24 producing two, 16 producing three, 12 producing four, and 14 producing five. Thus, 106 departments produced fewer and 52 departments more than the average. The 52 that produced more than the average — one third of the departments — produced 71 percent, or more than two thirds, of the PhDs.

The over-average group of 58 expects to increase its PhD production in one year far less (by 38, or 10 percent) than the under-average group (by 179, or 76 percent). Either the over-average group does not expect to expand much, compared to the under-average group, or it is far better at predicting PhD-production rates. In either case, it is clear that there will be little future growth in the over-average departments. In successive years since 1984-85, the predicted one-year growth by the under-average group was 167 percent, 164 percent, 116 percent, 82 percent, and 76 percent, respectively.

Sex and minority status. Table 3 gives the figures on PhDs awarded to minority students and women. The figures are unfavorable from the standpoint of minority and female representation in the field. This is the case despite the fact that the percentage of PhDs going to females did rise to 14 percent. Table 4 presents statistics for the 20-year period beginning in 1970, with the data before 1984-85 taken from Taulbee.[2] Throughout the 1980s, the percentage of PhDs who are women has remained relatively constant at about 11 percent, those who are black at 0.5 percent, and those who are Hispanic at 2 percent.

Citizenship. The number of PhDs given

Table 4. Sex, minority status, and citizenship of CS PhDs since 1970.

Yr.	Total	Women	PCT	Black	PCT	Hisp.	PCT	Foreign	PCT
70	112	1	1	1	1	—	—	22	20
71	124	4	3	1	1	—	—	21	17
72	206	12	6	2	1	—	—	39	19
73	208	7	3	2	1	—	—	41	20
74	203	6	3	2	1	—	—	46	23
75	256	21	8	1	0	—	—	68	27
76	246	14	6	0	0	—	—	57	23
77	208	14	7	0	0	—	—	68	33
78	223	19	9	2	1	—	—	51	23
79	248	24	10	1	0	—	—	65	26
80	230	28	12	0	0	—	—	82	36
81	235	26	11	0	0	—	—	79	33
82	244	27	11	1	0	—	—	83	34
83	256	31	12	2	1	—	—	86	34
84	274	29	10	3	1	—	—	87	32
84-85	326	32	10	3	1	7	2	122	37
85-86	412	50	12	6	1	6	1	184	45
86-87	466	51	11	1	0	8	2	181	40
87-88	577	60	10	4	1	5	1	238	41
88-89	625	87	14	0	0	6	1	248	40

Table 5. Employment of the PhDs.

	No. of PhDs	Unemployed	Self-Employed	Academia			Industry	Govt.	Outside US and Canada	Unknown
				PhD Dept.	Not PhD Dept.	Not CS or CE				
CS	625	9	7	229	63	17	181	24	56	39
PCT	—	1	1	—	—	49	29	4	9	6
CS-CE	807	10	7	259	65	26	227	26	62	125
PCT	—	1	1	—	—	43	28	3	8	15

Table 6. Undergraduate and masters degrees.

Non-PhD Degrees PhD Depts. Only	Undergraduate						Masters					
	84-85	85-86	86-87	87-88	88-89	89-90	84-85	85-86	86-87	87-88	88-89	89-90
CS No. degrees	10,422	10,947	10,540	10,759	8,796	9,037	2,889	3,720	3,614	4,150	4,297	4,444
No. Depts. Responding	96	116	121	127	120	120	101	116	123	127	129	129
Ave. Per Dept.	9	94	87	85	73	75	29	32	29	33	33	35
CE No. Degrees	—	—	2,103	1,928	1,810	1,776	—	—	731	1,009	1,160	1,180
No. Depts. Responding	—	—	22	30	26	27	—	—	22	30	29	29
Ave. Per Dept.	—	—	96	64	70	66	—	—	33	34	40	41
CS-CE No. Degrees	—	—	12,643	12,687	10,606	10,813	—	—	4,345	5,159	5,457	5,624
No. Depts. Responding	—	—	143	157	146	147	—	—	145	157	158	158
Ave. Per Dept.	—	—	88	81	73	74	—	—	30	33	35	36

Table 7. New graduate students in fall 1989.

	Total New Graduate Students	With CS Degrees	PhD Program	Masters-Only Program	Part-time Masters Students
CS Total	3,967	1,977	1,632	2,335	1,009
CS Depts. Responding	129	129	129	129	129
CS Ave. Per Dept.	31	15	13	18	8
CE Total	1,072	157	361	711	407
CE Depts. Responding	29	29	29	29	29
CE Ave. Per Dept.	37	5	12	25	14
CS-CE Total	5,039	2,134	1,993	3,046	1,416
CS-CE Depts. Responding	158	158	158	158	158
CS-CE Ave. Per Dept.	32	14	13	19	9

to foreigners (again, meaning non-Americans and non-Canadians) increased by 10, although the percentage actually declined one point. Table 4 contains figures for foreigners from 1970 to 1988.

Employment. As Table 5 shows, 49 percent took faculty positions in CS in the US or Canada (last year, the figure was 51 percent). There was little change from last year.

Undergraduate and masters degrees. Many universities and colleges have undergraduate and/or masters programs, but do not award the PhD. Therefore, the data given below says little about the field of computer science as a whole.

Table 6 provides statistics on undergraduate and masters degrees in PhD departments, with columns labeled "89-90" representing expectations. The CS departments reported a decrease of 18 percent in undergraduate degrees. In part, this was because several departments did not complete this portion of the survey, although the average decrease of 12 undergraduate degrees per department is substantial. The departments expect an increase again next year, but the decrease is still worrisome.

New graduate students in fall 1989. Table 7 gives enrollment figures for new students in fall 1989. In the table, "PhD Program" stands for the number of new graduate students in PhD programs, regardless of whether they intend to earn a masters degree first. The number of new graduate students in CS decreased 2 percent from last year (to 3,967 from 4,067), but the number of new graduate students in CS PhD programs rose 9 percent to 1,632 from 1,497.

Faculty data

Table 8 contains statistics on departmental faculty in January 1989. In this table, all figures are stated in terms of full-time equivalents. For example, two half-time appointments are counted as one full-time position.

CS, again, saw little change over the previous year in the proportions of faculty at the three levels. CS remains a relatively young field, with fewer full professors (7.0) than assistant professors (7.3) per depart-

Table 8. Faculty statistics, 1989-90 academic year.

Faculty	All CS-CE Depts. (157 of 158 depts.)		128 of 129 CS Depts.		Top 24 CS Depts.		Other (104 of 105 depts.)	
	Total	Ave.	Total	Ave.	Total	Ave.	Total	Ave.
Tenure-Track Faculty	3,240	20.6	2,550	19.9	689	28.7	1,861	17.9
Asst. Prof.	1,142	7.3	938	7.3	223	9.3	715	6.9
Assoc. Prof.	909	5.8	718	5.6	176	7.3	542	5.2
Full Prof.	1,189	7.6	894	7.0	290	12.1	604	5.8
Nonteaching Research Faculty	171	1.1	143	1.1	73	3.0	70	0.7
Postdoctorates	136	0.9	108	0.8	60	2.5	48	0.5
Nontenure-track Teachers	416	2.6	362	2.8	74	3.1	288	2.8
Other Faculty (e.g. visitors)	237	1.5	205	1.6	69	2.9	136	1.3

ment. However, we are seeing a gradual change as the field matures. The top 24 departments have a larger per-department number (12.1) of full professors than assistant professors (9.3).

Women and minorities. At the request of Nancy Leveson of the University of California at Irvine, we added a new question to the 1988-89 survey concerning the number of women, blacks, and Hispanics in the PhD-granting computing departments. Table 9 provides the bleak statistics derived from the question.

Table 9. Women and minorities on CS and CE faculties.

	Total	Women	PCT	Blacks	PCT	Hispanics	PCT
CS Asst. Prof.	938	92	10	6	1	20	2
CS Assoc. Prof.	718	66	9	2	0	6	1
CS Full Prof.	894	30	3	4	0	7	1
CS Total	2,550	188	7	12	0	33	1
CE Asst. Prof.	204	11	5	3	1	4	1
CE Assoc. Prof.	191	8	4	0	0	2	1
CE Full Prof.	295	3	1	1	0	1	0
CE Total	690	22	3	4	0	7	1

That the numbers of blacks and Hispanics in our faculties are so low was no surprise — since 1973, less than 1 percent of our PhDs have been black or Hispanic. Thus, you can't expect to have more than 1 percent of blacks and Hispanics as faculty members. Blacks and Hispanics must be attracted into our field at a much lower level — in our high schools and colleges.

For women, however, the results are gloomy in another way. The numbers are fine at the assistant-professor level in that the percentage of women (10 percent) is the same as the percentage in PhD production (10-12 percent since 1973 and 14 percent this year). But, at the associate- and full-professor levels, the percentages of women fall drastically to 9 percent and 3 percent, respectively, indicating that we are not retaining women as faculty members over the years. At the professorial level, there aren't enough women for one per department. In toto, there are scarcely 1.5 women per department. (Leveson has analyzed the plight of women in computing in detail in a report written for Erich Bloch, director of the NSF.)

Hiring for 1988-89. CS-CE departments reported hiring 259 new faculty, or 1.6 per department. CS departments in the US hired 180, or 1.4 per department. Salaries were reported for new PhDs hired for fall 1989 by 75 US CS-CE departments, 62 US CS departments, and 10 Canadian departments. Table 10 provides this salary information. The data for the Canadian universities is shown separately in Canadian dollars. It should be noted that Canadian salaries are on a 12-month scale, that Canadian and US dollars command different values, and that the typical amounts of consulting that can be performed in the two countries differ.

The average US salary for a new PhD increased from $36,668 in fall 1985 to $38,957 in fall 1986 (6.2 percent) to $40,885 in fall 1987 (4.9 percent) to $42,653 in fall 1988 (4.3 percent) to $44,946 in fall 1989 (5.0 percent).

Table 11 lists the number of CS-CE departments averaging salaries in each $1,000 range for fall 1989 and four previous years (the numbers are rounded and presented in thousands of dollars).

Faculty salaries. Table 12 summarizes nine-month faculty salaries in US CS and CS departments during the 1988-89 academic year. The second column of Tables

Table 10. New PhD salaries for fall 1989.

	All US CS-CE Depts.	All US CS Depts.	Top 24 US CS Depts.	Other 103 US CS Depts.	12 Canadian CS Depts.
Total Hired	259	180	43	137	24
No. Depts. Reporting Salaries	75	62	19	43	10
Minimum	$39,000	$39,000	$42,500	$39,000	$34,970
Average (of the averages)	$44,946	$44,985	$45,381	$44,810	$44,519
Maximum	$56,650	$56,650	$49,000	$56,650	$48,340

Table 11. Number of departments with new US PhD salaries 1985-86 through 1989-90.

Salaries*	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51+
1985-86	2	10	11	11	5	5	1	0	0	0	0	0	0	0	0	0	0
1986-87	3	1	9	11	16	14	5	4	2	0	0	0	0	0	0	0	0
1987-88	1	1	1	3	8	13	14	20	4	1	1	0	1	1	0	0	0
1988-89	0	0	0	2	2	5	13	3	19	11	14	0	2	1	0	0	0
1989-90	0	0	0	0	1	1	1	7	14	12	1	11	8	3	1	1	3

(* In thousands of dollars)

Table 12. Nine-month salaries, 135 of 146 US CS and CE departments.

Faculty Rank	No.	Reported Minimums			Average Over All	Reported Maximums		
		Min	Mean	Max		Min	Mean	Max
Asst.	993 (1,061)	$27,200	$42,762	$51,000	$45,634	$37,018	$48,970	$65,970
Assoc.	752 (809)	$28,300	$47,583	$61,000	$52,669	$44,035	$57,940	$85,000
Full	960 (1,070)	$36,000	$57,338	$83,040	$68,699	$51,000	$83,802	$135,000

Table 13. Nine-month salaries, 109 out of 117 US CS departments.

Faculty Rank	No.	Reported Minimums			Average Over All	Reported Maximums		
		Min	Mean	Max		Min	Mean	Max
Asst.	812 (857)	$27,200	$42,828	$49,000	$45,679	$37,018	$48,985	$57,800
Assoc.	594 (618)	$28,300	$47,848	$61,000	$53,094	$44,035	$58,430	$74,325
Full	717 (775)	$36,000	$57,996	$83,040	$69,326	$54,300	$84,236	$135,000

Table 14. Twelve-month salaries, 12 of 12 Canadian CS departments (in Canadian dollars).

Faculty Rank	No.	Reported Minimums			Average Over All	Reported Maximums		
		Min	Mean	Max		Min	Mean	Max
Asst.	81 (81)	$36,952	$45,999	$53,000	$49,178	$42,793	$52,188	$65,830
Assoc.	100 (100)	$42,930	$52,703	$64,202	$59,383	$51,376	$69,132	$107,000
Full	119 (119)	$56,885	$65,881	$73,000	$76,560	$73,904	$88,826	$115,104

Table 15. Desired faculty growth.

	1989-1990	1990-1991	1991-1992	1992-1993	1993-1994	1994-1995	5-Yr. Increase
CS Faculty Size	2,573	2,774	2,919	3,040	3,145	3,240	25 %
CS Ave. Size	20.1	22.5	23.6	25.6	24.0	25.3	—
CS-CE Faculty Size	3,281	3,521	3,698	3,844	3,965	4,075	24 %
CS-CE Ave. Size	21.0	22.4	23.5	24.5	25.2	26.0	—

12 to 14 gives the number of faculty (in each rank) for which salaries were reported and, in parentheses, the total number of faculty in that rank.

Departments reported the minimum, mean, and maximum salaries of assistant, associate, and full professors and the number of faculty in each rank. For minimum salaries (as well as maximum), the table shows the minimum, average, and maximum. The average is given over all salaries in each faculty rank. This is not the average of the means, but the true average.

Comparing this year's CS figures with last year's, we found that the average assistant professor's salary increased 4 percent from $43,959 to $45,679, the average associate professor's salary rose 5 percent from $50,806 to $53,094, and the average full professor's salary grew 3 percent from $67,205 to $69,326.

Forty-six US departments reported a maximum full-professor salary of greater than $90,000.

Table 14 gives salary information for the 12 Canadian departments.

Estimates of department growth. The departments were asked to estimate their faculty sizes through 1994-95, given an adequate supply of applicants. The 158 CS-CE departments would like to grow by 794 (24 percent) by 1994-95. The 128 CS departments would like to grow by 667 to 3,240 (25 faculty per department).

Last year, 127 CS departments reported a desire to grow from 2,477 (19.5 per department) to 2,684 (21.1 per department). However, this year they report growing less than half as much, to 20.1 per department. Except for the six largest depart-

Table 16. Faculty losses.

	CS-CE Depts.			CS Depts.		
	With PhD	Without PhD	Total	With PhD	Without PhD	Total
Died or Retired	22	7	29	14	4	18
Were Visitors, Returned to Employers	17	0	17	15	0	15
Teaching Elsewhere	76	4	80	63	4	67
Left for Nonacademic Positions	49	3	52	44	3	47
Returned to Graduate School	0	1	1	0	1	1
Other	21	2	23	12	1	13
Total	185	17	202	148	13	161

ments, Table 15 indicates that all departments desire roughly the same growth, regardless of size.

Faculty losses. Table 16 gives statistics on faculty losses. The CS departments reported losing 0.7 percent of the faculty through death and retirement; and the CE departments, 1.1 percent. We don't expect higher percentages of retirement in CS for another 5-10 years. Of the other 173 CS-CE faculty who left, at least 35 percent left for other teaching positions, 26 percent left academia, 13 percent were visitors who returned to their employers, and 3 percent returned to graduate schools. The percentages for CS were very similar: 35 percent, teaching elsewhere; 27 percent, positions outside academia; 14 percent, visitors; and 3 percent returned to graduate school. This year, 161 faculty left the departments; last year the figure was 179. ■

References

1. "An Assessment of Research-Doctorate Programs in the United States," National Academy Press, 1982.
2. O.E. Taulbee, "Annual US Summaries of PhD Production and Employment in Computer Science, 1970-1985," *SIGCSE Bulletin*, Vol. 18, No. 3, Sept. 1986, pp. 2-8, 12.
3. D. Gries, "The 1984-85 Taulbee Survey," *Comm. ACM*, Vol. 29, No. 10, Oct. 1986, pp. 972-977, and *Computer*, Vol. 19, No. 11, Nov. 1986, pp. 69-71.
4. D. Gries, "The 1985-86 Taulbee Survey," *Comm. ACM*, Vol. 30, No. 8, Aug. 1987, pp. 688-694.
5. D. Gries and D. Marsh, "The 1986-87 Taulbee Survey Report," *Comm. ACM*, Vol. 31, No. 8, Aug. 1988, pp. 984-991, and *Computer*, Vol. 21, No. 8, Aug. 1988, pp. 53-61.
6. D. Gries and D. Marsh, "The 1987-88 Taulbee Survey Report," *Comm. ACM*, Vol. 32, No. 10, Oct. 1989, pp. 1,217-1,224, and *Computer*, Vol. 22, No. 11, Nov. 1989, pp. 49-56.

David Gries has been at Cornell University since 1969, having served as Computer Science Department chair from 1982 to 1987. He received his doctorate from the Munich Institute of Technology in 1966, was a co-recipient of the ACM Programming Languages and Systems Best Paper Award in 1976, was a Guggenheim Fellow during 1984-85, and received the AFIPS Education Award in 1985.

Dorothy Marsh has been employed at Cornell University since 1984 and has been an administrative assistant in the university's Computer Science Department since 1986. She was previously employed by the US Department of Agriculture.

The address of the authors is Computer Science Department, Upson Hall, Cornell University, Ithaca, NY 14853-7501.

For a complete copy of this year's Taulbee Survey, including nine additional tables, see pp. 160-169 of the September 1990 edition of *Communications of the ACM*.

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

INFORMATION & COMPUTER SCIENCE DEPARTMENT

APPLICATIONS ARE INVITED FOR ASSOCIATE TO FULL PROFESSOR POSITIONS AS REGULAR FACULTY OR VISITING FACULTY. A PH.D. IN COMPUTER SCIENCE AND EVIDENCE OF RESEARCH ACCOMPLISHMENT ARE REQUIRED. POSITIONS ARE ALSO AVAILABLE FOR LECTURERS WITH M.Sc. IN COMPUTER SCIENCE AND A MINIMUM OF TWO YEARS TEACHING EXPERIENCE.

FACULTY WILL INTERACT WITH UNDERGRADUATE AND GRADUATE PROGRAMS, AND WILL HAVE FREE ACCESS TO EXTENSIVE LAB, COMPUTER AND LIBRARY FACILITIES. THE COMPUTING FACILITIES INCLUDE IBM 3090, SEVERAL VAX 11/780, MICRO VAXES, INTERGRAPH, DEC & VAX WORKSTATIONS RUNNING ON ETHERNET AND OTHER SMALLER MACHINES.

KFUPM offers attractive salaries commensurate with qualifications and experience, and benefits that include free furnished airconditioned accommodation on campus, yearly repatriation tickets, ten months duty each year with two months vacation salary. Minimum regular contract for two years, renewable.

Interested applicants are requested to send their Curriculum Vitae with supporting information not later than one month from the date of this publication, to:

DEAN OF FACULTY AND PERSONNEL AFFAIRS
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DEPT NO. 9022
DHAHRAN 31261, SAUDI ARABIA

Journal of INFORMATION PROCESSING

BY INFORMATION PROCESSING SOCIETY OF JAPAN

Journal of Information Processing (JIP) is a quarterly journal published since 1978 by the Information Processing Society of Japan (IPSJ). It publishes original research contributions, survey and review articles and English translations of selected papers originally published in other publications of IPSJ in a wide range of computer and information science and technology. Special issues present a variety of papers and articles describing activities in Japan for areas of special interest. Recent and planned special issues are as follows.

vol. 13 No. 1 (March 1990) Japanese Text Processing
- Collection of articles describing majour technical issues such as character set, human interface and Kana to Kanji conversion.

vol. 13 No. 2 (August 1990) 32 bit Microprocessors in Japan
- Collection of original contributions and review articles describing the details of original 32 bit microprocessors built by NEC, TOSHIBA, NTT, ICOT, ETL and Hitachi.

vol. 13 No. 3 (October 1990) Lisp Systems in Japan
- Collection of review articles on Lisp systems in Japan including TAO, Kyoto Common Lisp, UtiLisp, EusLisp, HLisp, HiLisp and MC/Lisp.

vol. 13 No. 4 (December 1990) Algorithms
- Selected research papers from international workshop on algorithms held in Kyushu, Japan in the fall of 1989.

vol. 14 No. 2 (July 1991) Software Engineering in Japan
- Collection of review articles on current focus, status and future perspectives from software research community and industries in Japan.

Subscription Order Form

Mail this to:
Japan Publication Trading Co., Ltd.
P.O. Box 5030 Tokyo International
Tokyo, Japan
Phone: 81-3-292-3753
Fax: 81-3-292-0410

Please enter / renew my subscription to the JOURNAL OF INFORMATION PROCESSING beginning with: Vol. ____ No. ____

Name: _____

Address: _____

_____

_____

Annual subscription rate for 1990:

by surface mail			... US$ 50.00
by air mail	Zone 1	Asia	... US$ 63.00
	Zone 2	North and Central America, Australia, New Zealand, Middle East	... US$ 66.50
	Zone 3	Europe and others	... US$ 70.00

☐ I enclose a check for US$ _____

☐ Charge to my credit card (local currency equivalent to the following Japanese Yen will be charged).

☐ by surface mail ...¥7,500

☐ by air mail ☐ Zone 1...¥9,400 ☐ Zone 2...¥9,900 ☐ Zone 3...¥10,500

☐ Visa ☐ Master Card

Card No. _____ Exp. Date _____

Signature _____

Editor: Fletcher J. Buckley, 103 Wexford Dr., Cherry Hill, NJ 08003, phone (609) 866-6350, fax (609) 866-7753, Compmail II, f.buckley

User interface management systems and application portability

Robert C. Seacord, Software Engineering Institute

Higher level programming languages and standard operating systems now provide greater portability of application software than previously possible. Software developed in C for Unix, for example, can be easily ported to a variety of different architectures and machines. Developing language and operating system standards such as ANSI C and IEEE Posix will further application portability.

At a quick glance, open systems are, at long last, apparently becoming a reality. But, is this really the case?

As porting software to different architectures becomes more and more a matter of simply recompiling the software for that architecture, a serious problem is apparent in portability with the user interface of these systems. Now, more than ever, a customer buying a computer platform is also buying a "look and feel" associated with that system.

When using an Apple Macintosh, for example, the user expects to be able to perform a variety of actions using a single-button mouse. When working with an MS-DOS application, the user expects to be able to perform the same actions using the keyboard. When running Open Look, Motif, or Next Step, the user expects the application to provide a defined look and feel. Porting an application to simply run on a different platform is insufficient; the application interface is required to behave much like it does with other applications developed for that environment.

Software designs are usually not extensible enough to allow the integration of different user interface toolkits, particularly if these toolkits employ significantly different models in their application interfaces. Changing toolkits or integrating new toolkits usually requires major application modification, which then requires extensive retesting of the application.

Current standards activities are just beginning to address this problem. To date, standards bodies have attempted to define an abstract user interface toolkit that can be implemented in different ways. Where this approach provides some degree of device independence, it does not allow for the removal of stylistic concerns from the application. For example, where one user interface style guide may call for a pull-down menu, another may call for a command line interface.

One approach for addressing this problem of application portability across multiple platforms is the definition and implementation of a method for separating the application from the user interface. This separation makes changing the user interface of the system practical and makes changing user interface toolkits without modifying the application software possible.

User interface management systems. The term user interface management system was coined at the 1982 Workshop on Graphical Input Interaction Technique (GIIT).[1] Among other things, UIMSs are intended to encourage the separation of a software system into an application portion and a user interface portion. The application portion of a system implements the core functionality, while the user interface portion implements the user interface dialogue.

UIMSs provide facilities for defining both presentation and the computer-human dialogue components of a user interface. A UIMS also may provide facilities to support prototyping, encourage a design that allows for easy modification of the user interface, support implementation and maintenance of the user interface, and allow for the incorporation of new user interface technologies.

Most UIMSs are based on the Seeheim architecture[2] (see Figure I). This architecture uses a layered approach similar to the one used in the International Standards Organization (ISO) Open Systems Interconnection (OSI) model. The architecture is intended to encourage the separation of functionality between the application and the user interface portions of a software system.

The three different layers of the architecture provide differing levels of control over user input and system output:

The *application layer* consists of the core functionality of the application that can be described in a presentation-independent manner. For example, in a calculator program, this would include the underlying math subroutines library.

The *dialogue layer* specifies the presentation-dependent portion of an application system including the dynamic behavior of the user interface. The dialogue should allow the display and removal of interaction objects without application involvement and support cascading menus, direct manipulation, and other user interface styles and techniques. The dialogue provides the mapping between the presentation and application layers. The user interface dialogue may be specified via a user interface definition language (UIDL) or by an interactive technique.

The *presentation layer* controls the end-user interactions and generates low-level feedback. The presentation layer consists of a collection of interaction objects defined for a given user interface technology.

Existing approaches. Since the 1982 GIIT Workshop, a number of efforts have been undertaken to build UIMSs

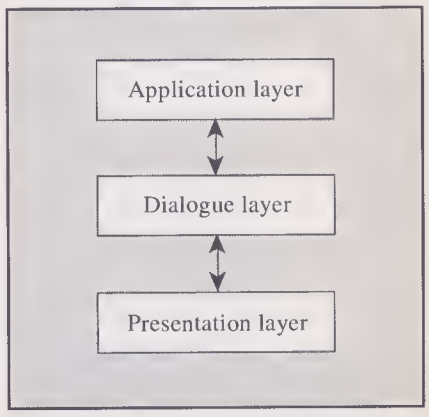


Figure 1. User interface management system architecture.

that achieve the goal of separation of concerns, while remaining a practical approach to software development. These systems have been classified by the model used in dialogue specification. Some of the more successful approaches have been

- event-driven,
- declarative,
- object-oriented,
- data-driven, and
- interactive layout systems.

In the event-driven approach, inputs are considered as events that are immediately handled by event handlers. Event handlers can cause output events, change the internal state of the system, and call application routines. Examples of event-driven systems include the University of Alberta UIMS,[3] ALGAE,[4] Sassafras,[5] and the Tele Use UIMS.[6]

Another approach is the use of declarative languages in which constraints are defined to specify what the system should do rather than how it should be done. An example of a system that takes this approach is Cousin.[7] In this category, a class of systems exists that automatically generates the user interface based on a definition of the semantic commands supported by the application. Examples are the UofA* UIMS[8] and MIKE.[9]

An object-oriented approach uses objects for defining user interactions, transforming data, and interacting with the application. A good example of a commercially available system that uses an object-oriented approach is Open Dialogue,[10] developed by Apollo Computer.

In the data-driven approach, the application communicates with the UIMS in terms of shared data elements. The UIMS behaves like an active database in that it provides a mapping between application data and user interface toolkit objects, and notifies the application of changes to application data resulting from user interactions. This approach was implemented in the Serpent UIMS[11] developed at the Software Engineering Institute at Carnegie Mellon University and the George Washington UIMS.[12]

Interactive layout systems allow the user to build a user interface by direct manipulation. Examples are Menulay,[13] Dialog Editor,[14] vu,[15] and TAE+.[16]

UIMS study group. The IEEE P1201 working group was formed in January 1989 and chartered to develop standards that would further application and user portability in the X Windows Environment.[17] Since P1201 was formed, the Open Software Foundation, Sun, and AT&T have independently developed toolkits for X Windows. Much of the P1201 effort has been spent trying to decide if any of these toolkits can serve as a basis for a standard or if a "virtual" toolkit approach can be used.

In August 1989, a UIMS study group was initiated in P1201 to determine if UIMS technology was sufficiently advanced to solve the problem of application portability across multiple platforms with different look-and-feel fields and to define the scope of a UIMS standard.

The group identified two components whose standardization would be beneficial to the industry. The first of these is an application programmers' interface that would

(1) provide a standard application programmers' interface across changes in the underlying toolkit;

(2) support the separation of an application into presentation-independent and presentation-dependent layers corresponding to the application, dialogue, and presentation layers of the Seeheim architecture; and

(3) allow the development of applications that are independent of presentation; for example, the underlying windowing system or user interface toolkit.

The second component is a UIMS interchange format. The purposes of a standard UIMS interchange format are to

(1) enable a wide variety of UIMSs to use a single format to store and exchange data, and

(2) allow vendors to develop compilers or interpreters that could "execute" the UIF on their platforms in a manner analogous to Postscript printers.

The P1201 UIMS study group has evaluated a number of user interface management systems including Serpent, Tele Use, and TAE+. The consensus of the group is that the state of the practice is sufficiently advanced to warrant a standards effort. It is believed that a UIMS standard would enhance both application portability and the state of the practice in user interface development.

References

1. J.J. Thomas and G. Hamlin, "Graphical Input Interaction Technique Workshop Summary," *Computer Graphics*, Vol. 17, No. 1, Jan. 1983, pp. 5-30.
2. M. Green, "Report on Dialogue Specification Tools," *User Interface Management Systems*, 1985, pp. 9-20.
3. M. Green, "A Survey of Three Dialogue Models," *ACM Trans. Graphics*, Vol. 5, No. 3, July 1986, pp. 244-275.
4. M.A. Flecchia and R.D. Bergeron, "Specifying Complex Dialogs in ALGAE," *Proc. SIGCHI+GI 87: Human Factors in Computing Systems*, ACM, Toronto, Apr. 1987, pp. 229-234.
5. R.D. Hill, "Event-Response Systems: A Technique for Specifying Multi-Threaded Dialogues," *Proc. SIGCHI+GI 87: Human Factors in Computing Systems*, ACM, Toronto, Apr. 1987, pp. 241-248.
6. Tele Logic, *Tele Use Reference Manual*, 1989.
7. P.J. Hayes, P.A. Szekely, and R.A. Lerner, "Design Alternatives for User Interface Management Systems Based on Experience with Cousin," *Proc. SIGCHI 85: Human Factors in Computing Systems*, ACM, San Francisco, Apr. 1985, pp. 169-175.
8. G. Singh and M. Green, "A High-Level User Interface Management System," *Proc. SIGCHI 89: Human Factors in Computing Systems*, ACM, New York City, 1989, pp. 133-138.
9. D.R. Olsen, "The Menu Interaction Kontrol Environment," *ACM Trans. Graphics*, Vol. 5, No. 3, 1986, pp. 318-344.

SDM ®

The SDM (System Development Methodology) ® is a widely used methodology to develop information systems in phases and activities. Tens of thousands of SDM books have been sold. SDM is the result of over 20 years of systems development experience. The present version is up to date with current practice and technology. CASE tools are used in conjunction with it. The phases are: Information Systems Planning, Definition Study, System Design, Detailed System Design, Implementation, Installation, Operation and Control. The SDM allows for flexible tailoring to projects and environments. Price US$96.00, 931 pages.

I hereby order___copies/more information of the SDM. I pay by ❒ American Express ❒ Visa.

Name:

Company:

Address:

City/State/Zip:

Telephone/Fax:

Card No: Expires:

Signature:

USA/Canada: 10427 Lawyers Rd., Vienna, VA 22181, Tel: (703) 255-2930, Fax:(703) 255-3602. EUROPE: Postbox 3027, 2280 GA Rijswijk, Netherlands. Tel: 31-70-3957235, Fax: 31-70-3191186.

CAP GEMINI PUBLISHING

Reader Service Number 7

10. Apollo, Inc., "Open Dialogue: Designing Portable, Extensible User Interfaces," 1987.

11. L.J. Bass et al., "Serpent: A User Interface Environment," *Proc. Winter 1990 Usenix Tech. Conf.,* Washington, DC, Jan. 1990, pp. 245-258.

12. J.L. Sibert, "An Object-Oriented User Interface Management System," *Proc. SIGGraph 86 Conf.: Computer Graphics*, ACM, Dallas, Aug. 1986, pp. 259-268.

13. W. Buxton et al., "Towards a Comprehensive User Interface Management System," *Proc. SIGGraph 83 Conf.: Computer Graphics,* ACM, Detroit, July 1987, pp. 35-42.

14. L. Cardelli, "Building User Interfaces by Direct Manipulation," *Proc. SIGGraph Symp. User Interface Software,* ACM, New York City, 1988, pp. 152-166.

15. G. Singh and M. Green, "Designing the Interface Designer's Interface," *Proc. SIGGraph Symp. User Interface Software*, ACM, New York City, 1988, pp. 109-116.

16. L. Szczur and P. Miller, "Transportable Applications Environment (TAE)+: Experiences in Objectively Modernizing a User Interface Environment," *Proc. OOPSLA 88,* San Diego, Calif., Nov. 1988, pp. 58-70.

17. S. Mehta, "User Interfaces and the IEEE P1201 Committee," *Unix Review*, Vol. 8, No. 1, Jan. 1990, pp. 14-20.

Nominations sought for two TAG posts

Nominations are being accepted for the 1991 chair and vice chair positions on the Technical Advisory Group of the Joint Technical Committee/Subcommittee 7, United States International Standards Organization/International Electrotechnical Commission. The US SC7 TAG coordinates the US position on international software engineering standards.

The names of interested and qualified candidates should be submitted to Bob Pritchard, IEEE Standards Office, 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331.

Nominees should have employer support to enable those elected to properly to carry out their TAG responsibilities.

The election will take place at the December 13-14 SC7 TAG meeting in San Diego, California.

The IEEE Computer Society provides administrative support to the TAG.

ISS

Singapore needs a brain wave.

No, we haven't run out of ideas. But we are looking for some bright sparks. A wave of the brightest minds in applied research, to take their place in arguably the most innovative work environment in Singapore.

The Institute of Systems Science. An institute with the ultimate task of improving the very fibre of Singapore's way of life.

We are currently doing work in Multimedia Systems, Hypermedia, Artificial Intelligence (including Neural Networks), Database, Distributed and Parallel Computing, Multilingual Systems, User Interface, Graphics and Image Processing and Communications.

We are looking for staff who have recently completed or will be completing a Ph.D., Masters or Bachelor's in computer science, electrical engineering, linguistics, cognitive science and related disciplines. (And if you're a Singaporean or come from Asia, all the more reason for you to consider coming home).

You'll work in a healthy research environment with the support of a team committed to developing world class research and operational prototypes.

You'll be stimulated by visiting world renowned scientists, our aesthetically designed work environment, and the latest hardware and software.

And of course, you'll be motivated by a competitive remuneration package that'll commensurate with your innovative output. To get the kind of people we want, we're willing to give what it takes.

If you're ready to take up this top-notch research opportunity, please send your resume to the Director of Personnel, National University of Singapore,10 Kent Ridge Crescent, Singapore 0511 or fax Singapore c/o ISS Recruitment Manager at 65-775-0938 or BITNET ISSAPPLY@NUSVM.

It may just be the start of a wave of rewarding challenges that your mind thrives on.

Moving?

Name (Please Print)

New Address

City State/Country Zip

PLEASE NOTIFY US 4 WEEKS IN ADVANCE

MAIL TO:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

ATTACH LABEL HERE

- This notice of address change will apply to all IEEE publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

Computer Society NEWS

Editor: Guylaine M. Pollock, Sandia National Laboratories, Division 1424, PO Box 5800, Albuquerque, NM 87185; phone, (505) 846-0040; Internet, gmpollo@sandia.gov

Computer Society chapter formed in Moscow

The most recent addition to the growing list of IEEE Computer Society chapters is the Moscow Chapter, which received official approval effective August 15, 1990. The interim chair is Nikolai V. Sazonov, pending election of officers.

An IEEE section, a prerequisite for the formation of a Computer Society chapter, was formed at the same time, with 65 members. "The simultaneous section and chapter formation is indicative of strong Soviet interest in computer technology," said Computer Society President Helen Wood, "and in the sharing of technical progress across former barriers."

The IEEE reports that Romania has successfully concluded a 13-year-long effort to form an IEEE section and that Czechoslovakia is also working to establish a section.

According to Michel Israel, chair of the Computer Society's European Activities Committee, the Moscow Chapter was created through the cooperative efforts of his committee, the USSR Ministry of Education, and the Popov Society (named for Aleksandr Stepanovich Popov, an early pioneer in the field of radio). Some of the top computer scientists in the USSR — academicians, directors of well-known institutes, and engineers — are included in the new chapter's membership, Israel said.

Benefits of Computer Society chapter status include participation in the Distinguished Visitors Program and the Chapters Tutorials Program, and support on conferences, workshops, and publications from both the Area Activities Board and the Computer Society staff.

The total number of Computer Society chapters now exceeds 130. In addition, the society has more than 120 student branch chapters. Besides the Moscow Chapter, new chapters have been formed during 1990 in El Salvador, Guatemala, Northern Italy, Indonesia, Venezuela, and Western Puerto Rico, and in Memphis and Kansas City in the US.

95 programs now accredited by Computer Science Accreditation Commission

Fourteen programs have been added to the accreditation list this year by the Computer Science Accreditation Commission, bringing the total to 95. The CSAC accredits baccalaureate computer science programs designed to prepare graduates for entry into the computer science profession. Programs accredited by CSAC meet or exceed the Computing Sciences Accreditation Board and the CSAC's criteria requirements for faculty, curriculum, laboratory and computing resources, students, and institutional support.

The accreditation process requires that the institution submit a comprehensive self-study of the program, and includes a campus visit by a CSAC evaluation team.

The Computing Sciences Accreditation Board was founded in 1984 by the ACM and the IEEE Computer Society in response to concerns about quality, consistency, and professional outcomes for graduates of computer science programs.

The 95 programs currently accredited by CSAC are listed below, followed by the year of initial accreditation. Unless otherwise stated, all programs are for the bachelor of science degree in computer science.

For more information about the accreditation process, including a copy of "CSAC/CSAB Criteria for Accrediting Programs in Computer Science in the United States," call or write to Patrick M. LaMalva, Executive Director, Computing Sciences Accreditation Board, 345 East 47th St., New York, NY 10017; phone (212) 705-7314, e-mail lamalva@um.cc.umich.edu.

Alabama
- Auburn University, Auburn (1987)
- University of Alabama, Huntsville (1988)
- University of Alabama, Tuscaloosa (1990)
- University of South Alabama (1988)
  - BS computer and information science; computer science specialization

Arkansas
- University of Arkansas, Little Rock (1990)

California
- California Polytechnic State University, San Luis Obispo (1986)
- California State University, Chico (1987)
  - Options: general; math/science; systems
- California State University, Fullerton (1988)
- California State University, Northridge (1987)
- California State University, Sacramento (1986)
- California State University, San Bernardino (1990)
- California State University, Stanislaus (1986)
- University of California, San Diego (1986)
- University of California, Santa Barbara (1986)
  - BS and BA computer science
- University of San Francisco (1987)
- University of Southern California (1988)
- University of the Pacific (1990)

Colorado
- United States Air Force Academy (1986)
- University of Colorado, Colorado Springs (1989)

Connecticut
Central Connecticut State University (1990)

District of Columbia
American University (1987)
George Washington University (1987)
Howard University (1988)
BS systems and computer science

Florida
Florida State University (1987)
University of Central Florida (1989)
University of North Florida (1987)
BS computer and information science; specialization in computer science
University of South Florida (1989)

Georgia
Georgia Institute of Technology (1986)
BS information and computer science

Illinois
Illinois Benedictine College (1989)

Indiana
Ball State University (1987)
BS or BA computer science (with or without co-op education)

Iowa
Iowa State University (1986)

Louisiana
Louisiana Tech University (1988)
Northeast Louisiana University (1987)
Southern University (1989)
Scientific option
Tulane University (1990)
University of New Orleans (1987)
University of Southwestern Louisiana (1987)

Maryland
Loyola College in Maryland (1990)
United States Naval Academy (1987)

Massachusetts
Brandeis University (1986)
BA computer science
Northeastern University (1986)
Southeastern Massachusetts University (1988)
University of Lowell (1990)
Worcester Polytechnic Institute (1986)

Michigan
Oakland University (1988)
Western Michigan University (1986)
Theory and analysis option

Minnesota
St. Cloud State University (1989)
University of Minnesota, Duluth (1989)

Missouri
Southwest Missouri State University (1989)
University of Missouri, Rolla (1986)

Mississippi
Mississippi State University (1986)
University of Mississippi (1990)
University of Southern Mississippi (1987)

New Hampshire
University of New Hampshire (1987)

New Mexico
New Mexico State University (1988)
University of New Mexico (1988)

New Jersey
Fairleigh Dickinson University (1987)
Teaneck Campus
New Jersey Institute of Technology (1986)
Stevens Institute of Technology (1986)

New York
Canisius College (1987)
College of Staten Island, CUNY (1989)
Pace University (1986)
Polytechnic University (1988)
Rochester Institute of Technology (1989)
State University of New York, Albany (1987)
State University of New York, Binghamton (1989)
State University of New York, Plattsburgh (1988)

North Carolina
Appalachian State University (1988)
North Carolina State University (1987)

North Dakota
North Dakota State University (1986)
University of North Dakota (1987)
BA/BS computer science

Ohio
Marietta College (1989)
Wright State University (1987)

Oklahoma
University of Tulsa (1988)

Pennsylvania
Allegheny College (1986)
Drexel University (1986)
Lehigh University (1987)
Program offered in College of Engineering and Physical Science
University of Scranton (1990)

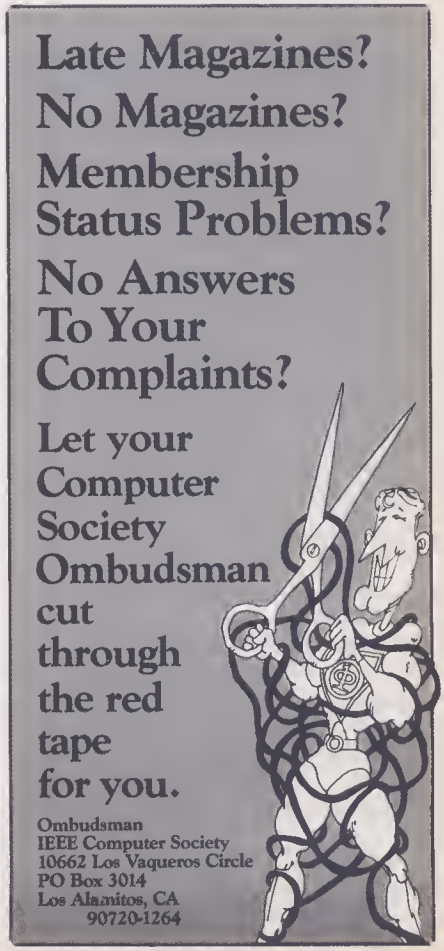
South Carolina
Clemson University (1986)
University of South Carolina (1990)
Winthrop College (1990)

Texas
Baylor University (1987)
Texas Christian University (1990)
University of Houston, University Park (1987)
University of North Texas (1986)
University of Texas, El Paso (1986)

Utah
Brigham Young University (1989)
University of Utah (1986)

Virginia
Hampton University (1989)
Old Dominion University (1990)
Virginia Commonwealth University (1988)

Washington
Eastern Washington University (1987)
Pacific Lutheran University (1989)
Western Washington University (1987)



Preliminary Conference Program
ACM SIGSOFT'90 – Fourth Symposium on Software Development Environments
3–5 December 1990, Irvine, California

Monday, December 3, 1990
Session 1 (9:00–10:00 a.m.) *Chaired by: Richard N. Taylor*

Invited Talk: Software Engineering Environments in the U.S. *by Barry Boehm, DARPA*

Session 2 (10:30–12:00 p.m.) *Chaired by: Susan L. Graham*

Low-Cost, Adaptable Tool Integration Policies for Integrated Environments *by David Garlan and Ehsan Ilias*
A Distributed Architecture for Programming Environments *by Dominique Clément*
Reconciling Environment Integration and Component Independence *by Kevin Sullivan and David Notkin*

Session 3 (1:30–3:00 p.m.) *Chaired by: Alexander Wolf*

Consistency Management in a Project Management Assistant *by Xiaolei Qian, Richard Jullig, and Marilyn Daum*
Configuration Management by Consensus: An Application of Law-Governed Systems *by Naftaly H. Minsky and David Rozenshtein*
Coping with Changes in an Object Management System Based on Attribute Grammars *by Lichao Tan, Yoichi Shinoda, and Takuya Katayama*

Session 4 (3:30–5:00 p.m.) *Chaired by: Peter Lee*

An Extensible Programming Environment for Modula-3 *by Mick Jordan*
The Pan Language-Based Editing System for Integrated Development Environments *by Robert A. Ballance, Susan L. Graham and Michael L. VanDeVanter*
A Multi-Formalism Specification Environment *by Edward A. Ipser, Jr.,David S. Wile and Dean Jacobs*

Tuesday, December 4, 1990
Session 5 (9:00–10:00 a.m.) *Chaired by: Dewayne Perry*

Invited Talk: Software Engineering Environments in Japan *by Michael Cusamano, Sloan School, MIT*

Session 6 (10:30–12:00 p.m.) *Chaired by: Koji Torii*

Fine Grained Data Management to Achieve Evolution Resilience in a Software Development Environment *by Richard Snodgrass and Karen Shannon*
A Procedure and Tools for Transition Engineering *by Abe Lockman and John Salasin*
Configuration Editing, Generation and Test within Working Contexts *by Hans-Ulrich Kobialka*

Session 7 (1:30–3:00 p.m.) *Chaired by: Gail Kaiser*

Reusing Software Developments *by Allen Goldberg*
An Environment for Documenting Software Features *by C. Frederick Hart and John J. Shilling*
A Program Integration Algorithm that Accommodates Semantics-Preserving Transformations *by Wuu Yang, Susan Horwitz, and Thomas Reps*

Session 8 & 9 (3:30–5:00 p.m.; 7:30–9:30 p.m.)

Demonstrations I & II

Wednesday, December 5, 1990
Session 10 (9:00–10:00 a.m.) *Chaired by: Richard Taylor*

Invited Talk: Software Engineering Environments in Europe *by Hubert Tardieu, SEMA MATRA, France*

Session 11 (10:30–11:30 a.m.)

Demonstrations III

Session 12 (1:00–2:30 p.m.) *Chaired by: Ian Thomas*

Software Process Enactment in Oikos *by V. Ambriola, P. Ciancarini, and C. Montangero*
Managing Software Processes in the Environment MELMAC *by Wolfgang Deiters and Volker Gruhn*
Language Constructs for Managing Change in Process-Centered Environments *by Stanley M. Sutton, Jr. and Dennis Heimbigner and Leon J. Osterweil*

Session 13 (3:00–4:30 p.m.) *Chaired by: Tony Wasserman*

Support for Change in RPDE^3 *by Harold Ossher and William Harrison*
Integrating Configuration Management into a Generic Environment *by Axel Mahler and Andreas Lampen*
A Query Service for a Software Engineering Database System *by Mohamed Tedjini, Ian Thomas, Guy Benoliel, Fernando Gallo, and Régis Minot*

Please make checks or money orders payable (in U.S. currency) to ACM SIGSOFT '90. Mail with completed registration form to

SIGSOFT '90 c/o Robin Sharp
Computer Science Room 458E
University of California at Irvine, Irvine CA 92717
(714) 725-2259 – rsharp@ics.uci.edu

Conference registration includes copy of proceedings, coffee breaks, Sunday night reception, and lunches Monday, Tuesday and Wednesday. The student fee does not include lunches. Refunds honored through November 23. Please circle the fee.

	before Nov 16	after Nov 16
ACM/SIGSOFT Member	$240	$270
Non-member	$270	$300
Full-time student	$90	$90

Name (for name tag): _____
Affiliation (for name tag): _____
Address: _____
_____
Phone: _____ Email: _____
ACM Membership Number: _____
Special meal req'ts – Kosher _____ Vegetarian _____ Other: _____

Irvine Hyatt-Regency Hotel*
17900 Jamboree Blvd, Irvine CA 92714
(714) 863-3111, Fax: (714) 852-1574

ACM SIGSOFT '90 Rates (not including tax) for each room (single or double occupancy). Please complete all information (type or print):

Room type	Rate	
Regular (November 30, December 1: $65):	$85	___
Concierge (November 30, December 1: $90):	$105	___
Student:	$74	___

Guest Name: _____
Address: _____
_____
Phone: _____
Arrival Date: _____ Departure Date: _____
Number in Party: _____ 1 bed rooms: _____ 2 Bed Rooms: _____
Reservations will be held until 6:00 pm on the date of arrival unless a one night deposit is received or is guaranteed with a major credit card.
Card Name: _____ Card Expiration Date: _____
Card Number: _____

Signature: _____

United Airlines is the official carrier of SIGSOFT '90, offering special fares to North American conferees: 5% saving off any published fare within the US/Canada, or 40% off full coach fare and valid from 27 November through 10 December 1990. call **(800) 521-4041** and cite **Plus ID No. 441HI**, or give this information to your local travel agent.

* Formerly the Irvine Hilton and Towers.

UPDATE

Contributions to Update are welcome. Send news of public policy or professional issues and of industrial or university research to *Computer*, Update, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264, or to Editor-in-Chief Bruce D. Shriver, Vice President for Research, University of Southwestern Louisiana, Drawer 42370, Lafayette, LA, 70504.

Microprocessor industry scrutinizes engineer's patent

Bob Carlson, Staff Editor

With potentially huge sums in royalties at stake, attorneys for Intel and other microprocessor manufacturers have been poring over the patent recently issued to Gilbert P. Hyatt of La Palma, California, for the invention of the microprocessor. Hyatt's successful bid for a patent on technology he claims to have devised in 1968 climaxed a 20-year effort by the engineer and inventor.

Hyatt, a member of the IEEE and ACM, says that he began work on the invention early in 1968 and built his first working computer later that year. He formed Micro Computer, Inc., the same year with the intent of developing the microcomputer technology. The company folded after about three years. Hyatt applied for the parent patent on December 28, 1970.

Intel actually produced the first commercial microprocessors the following year, but Hyatt never gave up pursuing the recognition he believed he deserved. "I needed financial support," Hyatt was quoted as saying in a Reuter wire service dispatch. "I suspect that's how my technology filtered down to the marketplace."

Millions of dollars in the balance. There has been much speculation but little in the way of specifics about what Hyatt might ultimately reap as a result of his patent. The *Washington Post*'s Evelyn Richards pointed out that Hyatt would not be able to claim infringement on any chips made before his patent was issued, but he could attempt to collect royalties on products sold during the next 17 years.

With so much at stake, a protracted David-and-Goliath struggle appears certain, and analysts seem reluctant to predict how the spoils are likely to be divided.

Hyatt, who has more than 50 other patents to his credit, works independently as an aerospace consultant, engineer, and inventor. He is optimistic about the present climate for innovation and has expressed a desire to plow his potential gains back into more high-tech R&D projects.

Lengthy inquiry predicted. Patent and copyright attorney Richard H. Stern, who writes a regular column in *IEEE Micro* on patent and copyright law, has not seen the patent but commented on the basis of published reports. Since Hyatt refiled several times with the patent office during his 20-year battle, Stern said, his applications will have to be examined carefully to make sure he wasn't simply reacting to ongoing progress in the field. "The concern is to what extent he kept adding things and making changes as he went along," Stern said. "It's not a free ride."

During the years that Hyatt spent making additional filings with the patent office, other significant patents were being issued for similar devices. So timing as well as the content of his applications becomes an issue in sorting out Hyatt's position. Most observers anticipate a lengthy and arduous investigation.

Challenge to the patent process. The 20-year time frame and the surprise that Hyatt's patent generated throughout the industry led one columnist to question the entire patent process and its ability to keep pace with the rapid growth of technology. "If even an abstract of Hyatt's application had been released 24 months after filing 21 years ago, semiconductor companies would have had an inkling of what was going on," wrote Michael Schrage in the *Los Angeles Times*. "They could have rushed their own applications or sought to cut a deal with the young inventor."

Schrage contrasted the US system, in which patent applications are sealed until the patent is awarded, with the procedure in Japan and Europe, where patent applications are made public 18 months after filing.

As the industry moves into months — perhaps years — of inquiries and legal maneuvers, one thing seems certain: A high-profile case such as this will turn up the volume in the ongoing discussion of how to establish, protect, and profit from intellectual property rights.

Research program seeks to improve CIM systems

The National Institute of Standards and Technology (NIST) Center for Manufacturing Engineering has joined the Applied Physics Laboratory of Johns Hopkins University in a one-year research program to achieve productivity and quality improvements in computer-integrated manufacturing systems. Surface-mount printed circuit boards are typical of the products targeted.

The project will include the specification of process and quality control techniques and strategies, computer hardware and software, data requirements, and interface and support standards such as the standard for the exchange of product model data and the Defense Dept.'s computer-aided acquisition and logistic support standards. In addition, new concepts for process control sensors will be tested.

The cooperative research project draws on NIST skill in process control sensors, automation, and standards technology, and the Applied Physics Lab's expertise in applications and electronics manufacturing.

IEEE takes position on computer crime

The United States Activities component of the IEEE has adopted a position on computer crime stating, in part, that existing legislation may not be clear and comprehensive enough to address all the problems of computer crime. Well-crafted federal legislation is recommended to promote uniform treatment of computer crime across the nation and provide significant deterrence as well as punishment.

According to the IEEE, such legislation should

- distinguish deliberately malicious acts from accidents;
- avoid imposing overly specific and burdensome operating requirements on computer systems operators, manufacturers, or users;
- cover a broad range of computer crimes and techniques and not be tool specific; and
- recognize trespass within an information system as a criminal act without requiring the system owner or operator to show that there has been damage or the potential to do damage.

Furthermore, the IEEE says that the law should not be written in a way that would discourage research on or discussion of the technology of malicious codes. Research into such techniques is an essential part of legitimate efforts to understand and control the threat of computer crime.

For more detailed information about the IEEE position on this issue, contact IEEE Public Relations, 1828 L St., N.W., Suite 1202, Washington, DC 20036-5104.

DoE shares risk-assessment methods

Security controls that cost more than the information they protect are not cost-effective. Thus, an important part of developing a computer security program is weighing the cost of controls against the risk of loss.

As part of its investigation into ways to identify risk and choose appropriate computer security measures, the National Institute of Standards and Technology (NIST) is offering information on methods developed by other federal agencies.

NIST recently reprinted a publication describing the risk assessment method used by the US Department of Energy. Steps specified in the DoE approach include

- defining the system, software, and data;
- identifying threats;
- selecting countermeasures; and
- obtaining management review, participation, and accountability.

The publication includes worksheets and tables needed to carry out each step, along with a sample completed work sheet, a bibliography, and a glossary.

US Department of Energy Risk Assessment Methodology (NISTIR 4325) is available from the National Technical Information Service, Springfield, VA 22161. Order by PB #90-244484 for $23 prepaid.

NSF raises graduate fellowship stipend

Students selected for awards in the 1991-1992 National Science Foundation graduate fellowship competition, conducted for NSF by the National Research Council, will receive stipends of $13,500 for a 12-month fellowship tenure. The cost-of-education allowance to the institution chosen by the fellow for graduate study will be $6,000 in lieu of all tuition costs and assessed fees. This year's competition will continue the special Women in Engineering component to encourage women to undertake graduate study in engineering fields.

NSF plans to award about 900 new three-year graduate fellowships to individuals who have demonstrated ability and special aptitude for advanced training in science or engineering. To increase the number of practicing scientists and engineers who are members of ethnic minority groups that have been underrepresented in the advanced levels of the nation's science and engineering talent pool, approximately 200 fellowships will be offered in the NSF minority graduate fellowship program.

The graduate fellowships are intended for students in the early stages of graduate study. Applicants are expected to take the Graduate Record Examination (GRE). The examinations will be given December 8, 1990, at designated centers throughout the United States and in certain other countries. NSF will pay the test fees for fellowship applicants, providing NSF application is the primary purpose.

The deadline for submission of applications to both the graduate fellowship program and the minority graduate fellowship program is November 9, 1990. Further information and application materials can be obtained from the Fellowship Office, National Research Council, 2101 Constitution Ave., Washington, DC 20418.

Parallel computing to tackle big problems

The University of California at San Diego and the San Diego Supercomputer Center have acquired a 32-processor iPSC/860 parallel computer from Intel Scientific Computers in Beaverton, Oregon. The iPSC/860 is built around Intel's i860 microprocessor, a RISC chip containing more than one million transistors.

Scientists from UCSD's Computer Science and Engineering Computing Facility will help create software for the system, as well as compilers capable of translating programs created for traditional computers for use on parallel systems.

The system, with a speed of nearly 2 billion floating-point operations per second, is a result of the Touchstone project, a three-year effort being carried out by Intel with the help of a $7.6 million grant from DARPA. The Touchstone project, launched in April 1989 with a total expected cost of $27.5 million, calls for a 2,048-processor machine with peak speeds of 150 gigaflops by 1992. The ultimate goal is a teraflop by 1995.

A report from the President's Office of Science and Technology Policy calls for a major jump in supercomputing performance to help solve some extremely difficult problems. These include world climate prediction, semiconductor circuit design and test, superconductivity research, mapping of the human genome, and support for advanced drug design.

Federal science officials have concluded that the best way to reach these scientific goals is through parallel supercomputers. The National Science Foundation is encouraging the four supercomputer centers it sponsors, one of which is SDSC, to become involved in parallel computing.

Preliminary Program

RTSS-90
Eleventh Real-Time Systems Symposium

December 4-7, 1990
Orlando, FL

Sponsors:
The Computer Society of the IEEE
TC Real-Time Systems

Scope: All aspects of the design, development, and analysis of real-time systems
General Chair: Dr. Doug Locke, IBM
General Program Chair: Professor Jane W.S. Liu, University of Illinois

Tuesday, December 4

Tutorial: "Rate-Montonic Scheduling Theory: Practical Applications",Lui Sha, Carnegie Melleon University and Doug Locke, IBM Federal Sector Div.

This tutorial focuses on the application of the rate monotonic scheduling (RMS) theory. The tutorial begins with an introduction to the theory, and addresses typical concerns regarding the practicality of the theory. For further information on the tutorial contact Linda Buss, address below.

Wednesday, December 5

On-site registration and check-in (coffee and pastries)
Welcome

Session 1: Scheduling and Resource Allocation I
- *Minimizing the Number of Late Tasks with Error Constraints,* J. Y-T. Leung, C.S. Wong
- *Resource Reclaiming in Real-Time,* C. Shen, K. Ramamritham, J. A. Stankovic
- *Scalability of a Distributed Real-Time Resource Center,* N. Griffeth, A. Weinrib

Session 2: Experimental Systems and Tools
- *Implementing a Verifier for Real-Time Systems,* D.A. Stuart
- *Experiments with a Program Timing Tool Based on Source-Level Timing Schema,* C.Y. Park, A. C. Shaw
- *From CHAOS ^{min} to CHAOS ^{arc}: A Family of Real-Time Kernels*, K. Schwan, A. Gheith, H. Zhou

Session 3: Concurrency Control and I/O Scheduling
- *Dynamic Real-Time Optimistic Concurrency Control* J. R. Haritsa, M. J. Carey, M. Livny
- *Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order,* Y. Lin S.H. Son
- *Scheduling I/O Requests with Deadlines a Performance Evaluation,* R. Abbott, H. Garcia-Molina

Thursday, December 6

Session 4: Communication
- *On-Line Routing of Real-Time Messages*, J. Y-T. Leung, T.W. Tam, G. H. Young
- *Analytic Evaluation of Contention Protocols Used for Real-Time Systems*, K.G. Shin, C. Hou
- *Real-Time Communication in Multiple Token Ring Networks*, Y-H. Lee and L-T. Shen

Session 5: Fault Tolerance
- *New Latency Bounds for Atomic Broadcast,* R. Strong, D. Dolev, F. Cristian
- *Agreeing on a Leader in Real-Time*, B.A. Coan, G. Thomas
- *A Robust Group Membership Algorithm for Distributed Real-Time Systems,* P. D. Ezhilchelvan, R. Lemos

Session 6: Operating System
- *The ARTS Real-Time Object Model*, C.W. Mercer, H. Tokuda
- *Time Capsules: An Abstraction for Access of Continuous-Media Data,* R. G. Herrtwich
- *Incremental Garbage Collection of Concurrent Objects,* D.M. Washabaugh and D. Kafura

Session 7: Scheduling and Resource Allocation II
- *The Preemptive Scheduling of Sporadic, Real-Time Tasks on One Processor,* S.K. Baruah, A.K. Mok, L.E. Rosier
- *A Stack-Based Resource Allocation Policy for Real-Time Processess,* T.P. Baker
- *Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines* J.P. Lehoczky

Special Session: Current Work

Short presentations: Symposium participants are invited to present their on-going work.

Friday, December 7

Session 8: Case Studies
- *Real-Time System Scenarios*, J. Molini, S. Maimon, P. Watson
- *The ATAMM Data-Flow Architecture,* S. Som, R.R. Mielke, J.W. Stoughton

Session 9: Language Support
- *Structuring Real-Time Systems with Performance Polymorphism,* K.B. Kenny, K.J. Lin
- *Applying Compiler Techniques to Scheduling in Real-Time Systems,* P. Gopinath, R. Gupta
- *Language Support for Maruti Real-Time System*, V.M. Nirkhe, S.K. Tripathi, A.K. Agrawala
- *MRL: A Real-Time Rule-Based Production System,* C-K. Wang, A. K. Mok, A.M.K. Cheng

Session 10: Semantics
- *A Calculus for Communicating Systems with Time and Probabilities,* H. Hansson, B. Jonsson
- *A Proof System for Communicating Shared Resources,* R. Gerber, I. Lee
- *Specifying and Verifying a Real-Time Priority Queue with Modal Algebra,* V. Yokaiken, K. Ramamritham

Session 11: Architecture
- *A Tagged Memory Technique for Recovery from Transient Errors in Fault-Tolerant Systems,* S.J. Adams, T. Sims
- *SMART (Strategic Memory Allocation of Real-Time) Cache Design,* D.B. Kirk, J.K. Strosnider
- *Real-Time Computing Supports in Futurebus+*, L. Sha, J. Lehoczky, R. Rajkumar

For hotel reservations contact
Walt Disney World Dolphin
1500 EPCOT Resort Blvd.
Lake Buena Vista, FL 32830
1-800-227-1500
(Be sure to request the conference rate: $130)
(Room rate only good until 11/18/90, after that date the price will increase.)

For more information contact:
Linda Buss
Dept. Mathematics/Computer Science
Macalester College
1600 Grand
St. Paul, MN 55105
(612)696-6287

Name _____

Organization _____

Address _____

City/State/Zip _____

Phone _____

IEEE Member # _____

Check here if you **do not** wish to have your name and address published in the list of attendees. _____

Mail registration forms to: 11th RTSS-90
c/o Linda Buss (address at left)

Registration Fee	Advanced (before 11-18-90)	Late (after 11-18-90)
IEEE Member	190	230
Non-Member	240	290
Full-Time Student	70	90
Tutorial		
Member	170	210
Non-Member	215	260

Total: _____

Make checks payable to IEEE 11th RTSS

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Editor: Richard Eckhouse, UMASS-Boston, Harbor Campus, Boston, MA 02125, Compmail II, r.eckhouse; Bitnet, eckhouse@umbsky; Compuserve, 70516,556

Postscript output on a Laserjet printer

T. L. (Frank) Pappas, AWEB Software Technology

Postscript output devices let you print a wide variety of typefaces in various sizes as well as graphics for producing books, reports, viewgraphs, brochures, etc. But what if you don't have a Postscript printer? These printers cost significantly more than HP Laserjet-compatible printers.

In this review, I look at three approaches to producing Postscript output on a Laserjet printer, using the Laserjet IIP. If you want more information about Postscript, the standard references are from Adobe Systems, published by Addison-Wesley: *The Postscript Language Reference Manual*, *Postscript Language Program Design*, and *The Postscript Language Tutorial and Cookbook*.

Postscript with TeX

In the August 1989 issue, I reviewed TeX on the IBM PC and described the standard Computer Modern fonts available with TeX. But there are also two TeX approaches for producing Postscript output.

If you're interested in TeX, look at *TeX for the Impatient* by Paul Abrahams et al., published by Addison-Wesley. It's both a good introduction to TeX and a good reference, so even experienced TeX users will like it. It won't replace Donald Knuth's *The TeX Book* (published by Addison-Wesley) as the final authority, but it's enough for most people. For $10, you can get a diskette containing the well-documented macros used in the book.

PC TeX 3.0

Knuth recently made some upward-compatible enhancements to TeX — making it version 3.0 — and is urging everyone to upgrade. Typical users won't use most of the enhancements, but macro-package writers might. For example, there's now a way to control the error messages TeX produces, so that errors can be described in terms of the macro package.

For users writing in languages other than English, the enhanced hyphenation features and 8-bit character support will make TeX better suited to their needs. TeX's ligature support has also been improved.

PC TeX, from Personal TeX, is one of the first PC versions to support version 3.0. This outstanding product comes in three flavors. PC TeX ($249) runs on an XT, AT/286, or AT/386. PC TeX/386 ($295) runs on any AT/386 with at least 2 Mbytes of memory. Big PC TeX/386 ($349) runs on any AT/386 with at least 4 Mbytes of memory.

Capacity limits of earlier versions have increased. For example, font memory has increased by 28 percent for all three versions. Page and memory capacity has quadrupled for Big PC TeX/386 and doubled for the other two. The maximum number of TeX commands has increased by 233 percent for Big PC TeX/386 and by 66 percent for the other two.

To give you an idea of relative speeds, I ran two chapters (43 pages) of a book I'm writing through the three versions. Big PC TeX/386 produced the .dvi file in 1 minute, 20 seconds, PC TeX/386 in 1 minute, 52 seconds, and PC TeX in more than 5 minutes.

In TeX, external fonts are related to internal names through font declarations. For example, "/tenrm=cmr10" associates the name "tenrm" with Computer Modern Roman at 10 points. Fortunately, TeX' style files contain definitions for commonly used fonts.

Figure 1 shows Postscript output produced with PC TeX. To use New Century Schoolbook, I changed the font declarations, as in "/tenrm=psmncsr." Here, "psmncsr" is the name of the Postscript New Century Schoolbook Roman font used by Arbortext's Postscript driver.

With the number of font names typically used in TeX — LaTeX has predefined about 50 — redefining each one would be annoying. Nelson Beebe, president of the TeX Users Group, sent me a preliminary collection of style files that do the renaming for you. Including the line "/input psmncs" in a TeX or LaTeX document redefines these names. (You can reach the TeX Users Group at PO Box 9506, Providence, RI 02940-9506.)

PC TeX is the best version of TeX for the PC. If you have a 386 and the necessary memory, you'll be better off buying PC TeX/386 or Big PC TeX/386. Personal TeX is at 12 Madrona Ave., Mill Valley, CA 94941, phone (415) 388-8853.

Reader Service 21

DVI Laser/PS

TeX creates a device-independent .dvi file that is printed or viewed using a driver program (I detailed this process in the August 1989 issue). For Postscript output, DVI Laser/PS, from Arbortext, is the driver I like.

Figure 1 shows that a TeX document can include text and Postscript graphics if the driver supports it. It also illustrates that different Postscript fonts can be used in the same document.

In many ways, DVI Laser/PS is like DVI Laser/HP for the Laserjet printers, which I reviewed in the August issue. In fact, it's the one I use when I print TeX documents on a Laserjet. In this review, I'll just concentrate on a few of the Postscript support features.

Postscript graphics are inserted via the "/special" command. DVI Laser/PS's version has several subcommands: epsfile inserts encapsulated Postscript files; plotfile inserts non-EPS files, such as the palette in Figure 1; overlay prints

a border, letterhead, etc. on each page of output; rotate lets you rotate a TeX box through a given angle; and revbar and revpoint draw revision bars in the margin.

The PC version of DVI Laser/PS sells for $225. It's also available for other computer systems. If you want to use TeX with Postscript output, get DVI Laser/PS. Arbortext is at 535 W. William St., No. 300, Ann Arbor, MI 48103, phone (313) 996-3566.

Reader Service 22

AP-TeX fonts

One problem with using Postscript with TeX is the reduced portability of generated documents. With Computer Modern fonts you can pass a TeX source file to any TeX user and get the same output regardless of the TeX version or printer being used. With some exceptions, the same is true for the .dvi file. Introducing Postscript removes this portability.

AP-TeX fonts from Kinch Computer, helps solve this problem. TeX uses .tfm font metric files to produce a .dvi file; a driver uses .pk font files to produce a file you can send to a printer. The AP-TeX fonts are a collection of .tfm and .pk files that provide Postscript-equivalent fonts in many sizes. In contrast, DVI Laser/PS uses .tfm files but produces a Postscript file.

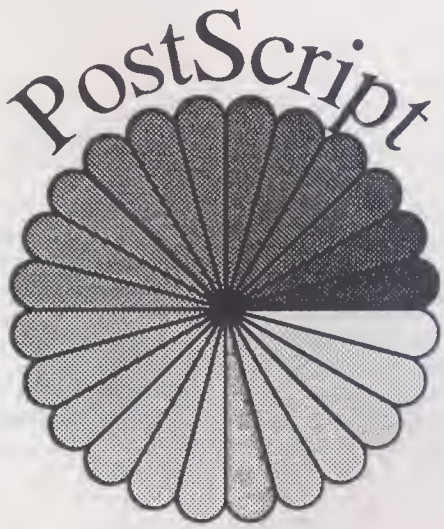
AP-TeX supports the 35 Postscript fonts found on an Apple Laserwriter. They are provided in 5, 6, 9, 10, 11, 12, 14.1, 17.3, 20.7, and 24.9 point sizes. Helvetica Bold, Palatino Bold, and Times Bold also have headlines sizes of 29.9, 35.8, 43.0, 51.6, 61.9, and 74.3 points. Even though only a fixed number of sizes are available, they should be enough for most uses.

There are some limitations when using the AP-TeX fonts. Figure 2 is the text of Figure 1 produced with AP-TeX on the Laserjet IIP. The palette is not included, since Postscript graphics cannot be produced using the fonts. However, the palette can be included by using a Postscript-to-PCL translator and including the PCL version using the "/special" command. Also note that the text in Figure 1 looks a little better — for example, look at "written" in the Times Roman paragraphs.

If you want Postscript capabilities with TeX while keeping some portability, the AP-TeX fonts may be for you. For $200, they are worth it. It would be nice if these fonts were as widely available as the Computer Modern fonts. You can reach Kinch Computer at 501 S. Meadow St., Ithaca, NY 14850, phone (607) 273-0222.

Reader Service 23

The palette of shades of gray below is taken from the PostScript Language Reference Manual. Text in this paragraph is written in Computer Modern 10-point. **Here is some bold text.**



Text in this paragraph is written in the PostScript font Avant Garde Book for 10-point roman, and Avant Garde Demi for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font New Century SchoolBook Roman for 10-point roman, and New Century Schoolbook Bold for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Times Roman for 10-point roman, and Times Bold for 10-point bold. **Here is some bold text.**

Figure 1. Postscript output from PC TeX using DVI Laser/PS.

Text in this paragraph is written in Computer Modern 10-point. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Avant Garde Book for 10-point roman, and Avant Garde Demi for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font New Century SchoolBook Roman for 10-point roman, and New Century Schoolbook Bold for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Times Roman for 10-point roman, and Times Bold for 10-point bold. **Here is some bold text.**

Figure 2. AP-TeX fonts from the HP Laserjet IIP.

Postscript in software

Now I'll look at some software products that let you print Postscript output on a Laserjet-compatible printer. I discuss hardware and software performance issues later in the section titled "Postscript in hardware."

Freedom of Press

Freedom of Press lets you produce Postscript output on a wide variety of non-Postscript printers. This software Postscript interpreter is from Custom Applications, 900 Technology Park Drive, Bldg. 8, Billerica, MA 01821, phone

(800) 873-4367. It provides the 35 scalable fonts found on an Apple Laserwriter and interprets Postscript graphics.

I produced Figure 3 using Freedom of Press on a Laserjet IIP. This may be a good substitute if you don't have a Postscript printer, but you can see problems in the output, particularly in Figure 3's last paragraph in "10-point roman"; the letters in "roman" are squeezed a little bit too close together, and the letters "r," "m," and "n" have stubs at the top.

Freedom of Press can be run as a resident program so that an application can write directly to a Postscript printer, with Freedom of Press capturing its output and printing directly to your printer. For programs like DVI Laser/PS that create a Postscript file, Freedom of Press can be invoked at the DOS command line.

In addition to the actual printers that Freedom of Press supports, it includes a special "printer" named PCX-File-200 that lets the program create 200-dpi PCX files.

As a software solution for Postscript output, Freedom of Press would make good sense at $495, but its poor performance (detailed below) makes it useless.

Reader Service 24

Goscript Plus

This software Postscript interpreter from Lasergo supports the same fonts and graphics as Freedom of Press but is not nearly as reliable. For example, I couldn't print Figure 1 using Goscript. When I tried, I got two error message from the interpreter. The first, "Current point not defined," occurred for each .dvi file I tried, but normally it would let me continue. The second, "Bad operand type," made the interpreter fail. To make matters worse, it left behind two of its temporary files.

I sent a copy of the Postscript output for Figure 1 to Lasergo, so maybe they will fix it. I used the new version (version 3), but the problem also occurs in version 2. I hope the product's reliability gets better, because it has a number of nice features. For example, Goscript supports a Postscript viewer that displays a full page of text on the screen so you can get an idea of what the final output will look like. This is especially useful for placing graphics such as the palette of Figure 1.

I was able to produce a version of Figure 1 by staying with the Computer Modern fonts. When I displayed the output on the screen, everything was properly placed, but only the upper half of the palette was filled in. The outline of each petal was visible, but the petals in the bottom half were filled in white. However, when Goscript produced output for the Laserjet, the palette looked fine.

For $299, this would be a really nice product if it were more reliable. It supports a wide range of printers and is considerably faster than Freedom of Press. You may be able to use it successfully with your products, but unless Lasergo will let you return it if it doesn't work with your applications, I'd wait for the next release. Lasergo is at 9235 Trade Place, Suite A, San Diego, CA 92126, phone (800) 451-0088.

Reader Service 25

Hijaak PS

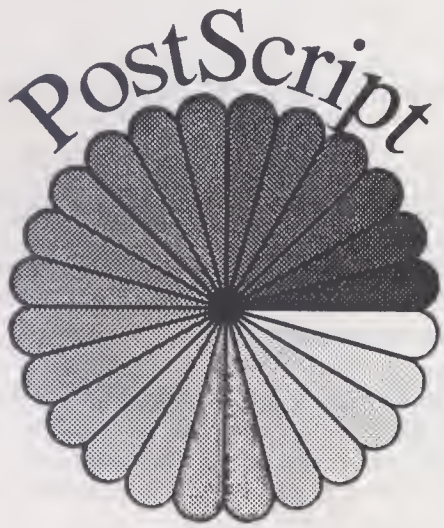
In the August 1989 issue I also reviewed Hijaak from Inset Systems. Hijaak is a program that converts between various file formats. Inset has teamed up with Lasergo for a special version of Hi-Jaak, called Hijaak PS, that converts Postscript files to the formats used by PC fax cards, such as JT Fax, The Complete Fax, and many others.

Hijaak PS, for $249, includes version 1.1 C of Hijaak and version 2.20 of Goscript (which only has 13 fonts). For $150 you can upgrade to version 3.0 of Goscript Plus. With the upgrade, you get Postscript output for your printer or fax, and Hijaak's great file-conversion capability. The one negative feature of Hijaak PS is Goscript's unreliability. However, once that problem is corrected, this hybrid product will be an excellent buy.

You can contact Inset Systems at 71 Commerce Drive, Brookfield, CT 06804, phone (800) 828-8088.

Reader Service 26

The palette of shades of gray below is taken from the PostScript Language Reference Manual. Text in this paragraph is written in Computer Modern 10-point. **Here is some bold text.**



Text in this paragraph is written in the PostScript font Avant Garde Book for 10-point roman, and Avant Garde Demi for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font New Century SchoolBook Roman for 10-point roman, and New Century Schoolbook Bold for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Times Roman for 10-point roman, and Times Bold for 10-point bold. **Here is some bold text.**

Figure 3. Postscript output using Freedom of Press.

Postscript in hardware

Another way to turn your Laserjet printer into a Postscript printer is to use a Postscript emulation cartridge that plugs into one of your printer's font-cartridge slots. The Pacific Page Personal Edition cartridge from Pacific Data Products is for the Laserjet IIP, but the company also makes cartridges for other Laserjet printers.

Figure 1 was produced using Pacific Page. One thing I really like about this cartridge is the consistent quality of the output and its reliability. I tried several complicated graphics, all of which printed flawlessly. Pacific Page also perfectly emulated the 35 scalable Postscript fonts.

The cartridge works by executing code out of a cartridge slot. The code is the Phoenix Page Postscript Interpreter from Phoenix Technologies. To use this cartridge you need 2.5 Mbytes of printer memory — the .5 Mbytes that's standard with the IIP and 2 Mbytes more.

I used Pacific's 2 Plus 2 memory board for the additional memory. It was easy to put in, but its construction prevented me from plugging in my generic 1-Mbyte memory board. However, Pacific claims that it will work with most other memory boards. The 2 Plus 2, which can also be used in a Laserjet III, lists for $595. It consists of 16 1-Mbit 120-ns DRAMs.

I also have Pacific's 25 in One font cartridge that provides 172 fonts and symbols sets compatible with 25 Laserjet cartridges. The 25 in One can be used in a Laserjet II, IID, IIP, or III. With the 25 in One you don't have to change cartridges to use different fonts. This is especially important with the IIP, which has only one cartridge slot. (The IIP's single slot also prevents you from using both Pacific cartridges at once, but that's Hewlett-Packard's design decision, not Pacific's, matched only by HP's soaking you $195 for a paper-feed tray.) Pacific also supplies drivers for such popular products as Windows, Pagemaker, Word, Wordperfect, Lotus 1-2-3, Excel, and Ventura Publisher. For $499, this cartridge is a must if you use applications that produce Laserjet output.

To get an idea of Pacific Page's performance, I ran some tests, which are summarized in Table 1. For the mixed graphics of Figure 1, Pacific Page and Freedom of Press each took about 2.5 minutes to print one page, while Goscript failed. But when I created a TeX file that repeated this figure on four pages, the last three pages printed much faster than the first.

To get full pages of text only, I removed the figures and extra white space from a copy of this review and used DVI Laser/PS to print first one page, then four pages. Pacific Page's numbers aren't too different from the first test, but Freedom of Press' results are horrible. Goscript took twice as long as Pacific Page in both text cases, which is OK for software emulation.

I then contrasted Pacific Page's performance with that of the IIP by running the text case in native PCL mode. Pacific Page took twice as long to print one page as native PCL mode, but 3.3 times as long for four pages. Since Postscript requires considerably more processing, it's not surprising that PCL native mode was faster. If the printer had 4 Mbytes of memory, I doubt the ratio would be as high.

You can switch between PCL and Postscript modes. The 25 in One can't be used in PCL mode, since the printer must be turned off before removing or inserting the Pacific Page cartridge, but you can switch modes without removing the cartridge.

At $499, Pacific Page is fast enough, especially if you can't afford a Postscript printer. Pacific Data Products is at 9125 Rehco Road, San Diego, CA 92121, phone (619) 552-0880.

Pacific Page: Reader Service 27
25 in One: Reader Service 28
2 Plus 2: Reader Service 29

Table 1. Postscript-emulation performance.

	Pacific Page	Freedom of Press	Goscript	PCL
mixed(1)	2:24	2:28	Failed	
mixed(4)	4:45	6:08	Failed	
text(1)	2:11	10:01	4:12	0:41
text(4)	5:08	33:11	10:57	1:24

Including illustrations

Once you have Postscript capability you'll want to include illustrations in your documents. You might want to use a scanner to get the illustrations, or you might want to use a drawing program to create them.

The Complete Page Scanner

In keeping with the low cost of the Postscript alternatives already reviewed, I looked at a low-cost scanner from The Complete PC (521 Cottonwood Drive, Milpitas, CA 95035, phone (408) 434-0145). For $899, The Complete Page Scanner is definitely complete.

The 13×9.5-inch scanner uses sheet feeding rather than a flat bed. It can scan images up to 8.5×14 inches and can scan at 200 or 300 dots per inch. Black-and-white or halftone imaging is provided. The latter supports four or eight shades of gray and can use either Bayer, mesh, or spiral dithering. Settings are selected from the front control panel.

Smartscan, the software provided with the scanner, simplifies much of the scanning and image use, rivaling software I've seen with more powerful scanners. For example, you can easily merge scanned images, as I did in creating Figure 4. First, I scanned Figure 1 as black-and-white text and used Smartscan's cropping and erasing functions to keep the text and palette in separate images. Next, I wrote the text that appears below the palette on a piece of paper and scanned it as a black-and-white image.

Once I had the three images, I used the Page function to paste them together. Using a mouse, I moved each image where I wanted it and hit the F10 key to freeze the image. Before positioning the palette and handwritten text, I used the Rotate function to rotate the images. The handwritten text was too big for the remaining space, so I used the Size function to reduce it.

While the scanned text looks pretty good, the palette isn't as good. I purposely picked an example that would show you how the scanner handles the worst case, an image in which adjacent sections do not contrast much. This lack of contrast is particularly noticeable in the upper right of the palette. When scanning images with reasonable contrast, the scanner produced reasonable results, especially for an $899 scanner.

The palette of shades of gray below is taken from the PostScript Language Reference Manual. Text in this paragraph is written in Computer Modern 10-point. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Avant Garde Book for 10-point roman, and Avant Garde Demi for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font New Century SchoolBook Roman for 10-point roman, and New Century Schoolbook Bold for 10-point bold. **Here is some bold text.**

Text in this paragraph is written in the PostScript font Times Roman for 10-point roman, and Times Bold for 10-point bold. **Here is some bold text.**

Figure 4. A version of Figure 1 scanned and altered using The Complete Scanner and Smartscan software.

Smartscan offers several other capabilities that make it very useful — such as image scaling from 10-200 percent, printing scanned images, and pixel editing of scanned images — but the one that really stands out is its conversion capability.

Scanned images can be converted to standard graphics formats: CUT, MSP, IMG, PCX, TIFF, and both high- and low-resolution formats for The Complete Fax. With the exception of TIFF, these formats can also be converted to the format used by Smartscan. Moreover, ASCII text and Epson FX graphics can be included. Smartscan also allows you to import files created by most popular word processors.

I've used the scanner to scan my company letterhead and include it on the cover page of every fax that I send. I have a JT Fax (which is now made by Hayes), so I converted the scanned image to PCX and used the JT Fax software to include it in the cover page.

To convert a scanned image to Postscript, I convert it to PCX and use Hijaak to convert it to Postscript. This lets me include scanned images in my Postscript documents.

I like this compact scanner, even though I prefer flat bed scanners to those with sheet feeders. But for $899, this scanner is enough for most people's needs.

Reader Service 30

Adobe Illustrator

Another way to include Postscript graphics in your Postscript documents is to use a drawing program such as Adobe Illustrator ($495). This fine program provides all the tools you need to create great graphics. A companion product, Adobe Streamline ($395) converts bitmapped images in PCX, TIFF, or PNT (Macpaint) format to Postscript for Illustrator, or to the formats for Corel Draw or Micrografx Designer.

Both products require Windows/286 or 386, although they come with a runtime version. A mouse is required. While a Postscript printer is recommended (one of the hardware or software interpreters already reviewed will do), you can use a Laserjet compatible to get draft quality output.

I really like Illustrator and Streamline, but before I tell you why, I want to mention a few problems. Illustrator does not yet execute under Windows 3.0. Adobe is working on a 3.0 version, but no release date has been announced. Adobe does say that the retail price of the Windows 3.0 version will be the same as the current version.

Under Windows/286 (and the runtime version), you must use an expanded memory manager. I have one that lets me use extended memory as expanded memory, but it doesn't work with Windows.

I also experienced some performance problems with Windows/386. I called Adobe for some advice, but what they told me to do just made things worse. Adobe promised to get back to me after my second call, but they never did.

The installation of both products should be changed. Illustrator tries to take you through all possible Windows installations in one set of combined instructions, forcing you to skip around sections that don't apply. This creates a maze of instructions that makes it easy to miss information. I had to redo the entire installation because I missed something the first time around. A separate set of instructions for each would have been more reasonable.

Both Illustrator and Streamline personalize the installation by asking for your name and organization. Incredibly, Illustrator modifies the distribution diskette with this information. Streamline doesn't do this, but it won't accept a long organization name — the name of my company is too long for Streamline but fine for Illustrator.

Now for the features. Figures 5 and 6 give a good idea of what you can create with Illustrator and Streamline and why I like them so much. There's more to these products than I could describe if I used the entire column, so I'll just describe how I created the figures.

For Figure 5, I simulated working with a scanned file by taking a TIFF file of a left eye (provided with Streamline) and converting it to a Postscript format (called AI by Adobe). Once I created the AI file, I read it into Illustrator, copied it to the clipboard, and pasted it into a document file. Clicking on the reflection tool brought up a dialog box that let me copy a reflection of the image around the vertical axis. (The toolbox look is typical of paint and drawing programs.)

When converting a file with Streamline, you have several options at your

disposal. Three basic methods are the outline method, the centerline method, and a combination of the two. The outline method traces an outline of each element in the image, filling each outline with either black or white. Adobe recommends this method for images with inconsistent line weights and filled areas, such as graphic designs and nontechnical line-art images. An image of a zebra is a good example.

The centerline method locates the center of each line and creates a stroked path. Adobe recommends this method for converting technical images that include consistent line weights and no filled areas, such as mechanical drawings and technical illustrations. The staircase in Figure 6 was converted this way from a TIFF file provide with Streamline.

I used the combination method to convert the TIFF file for the eye. This method is for images that contain both consistent line weights and filled areas.

Each method has options to enhance the conversion process. For example, both the outline and combination methods let you set a noise level to eliminate stray pixels that result from scanning a poor-quality image. Streamline ignores any pixel areas smaller than the noise level. Also, you can use the centerline and combination methods to make all lines a uniform width of 0-12 points. If the line widths in the image are not uniform, you can let Streamline try to match the line widths in the bitmapped image.

Once you have converted a bitmapped image for Illustrator, you can modify it in many ways. I already described reflection. It's also easy to scale images. The eye in Figure 6 was scaled down from its original size (as shown in Figure 5). The staircase was also scaled down. As with most Illustrator capabilities, there's a quick way and a precise way to do things. I'll explain the scaling tool in detail so you'll see how easy the tools are to use.

After selecting the scaling tool, you can quickly scale an image by clicking on a point to the left and below the image (the selection point), moving the mouse to a point above and to the right of the image, and then dragging the mouse away to increase the size, or dragging it towards the image to decrease the size. Holding down the Shift key results in uniform scaling; holding down the Alt key results in a scaled copy of the image.

For precise scaling, press the Alt key at the selection point. This brings up a dialog box that lets you specify the exact scaling. If you want nonuniform scaling, you can specify the horizontal and vertical scaling.

Excluding the eye, staircase, and text, the other images in Figure 6 are supplied

Figure 5. A **TIFF file, duplicated and "reflected" in Adobe Illustrator.**

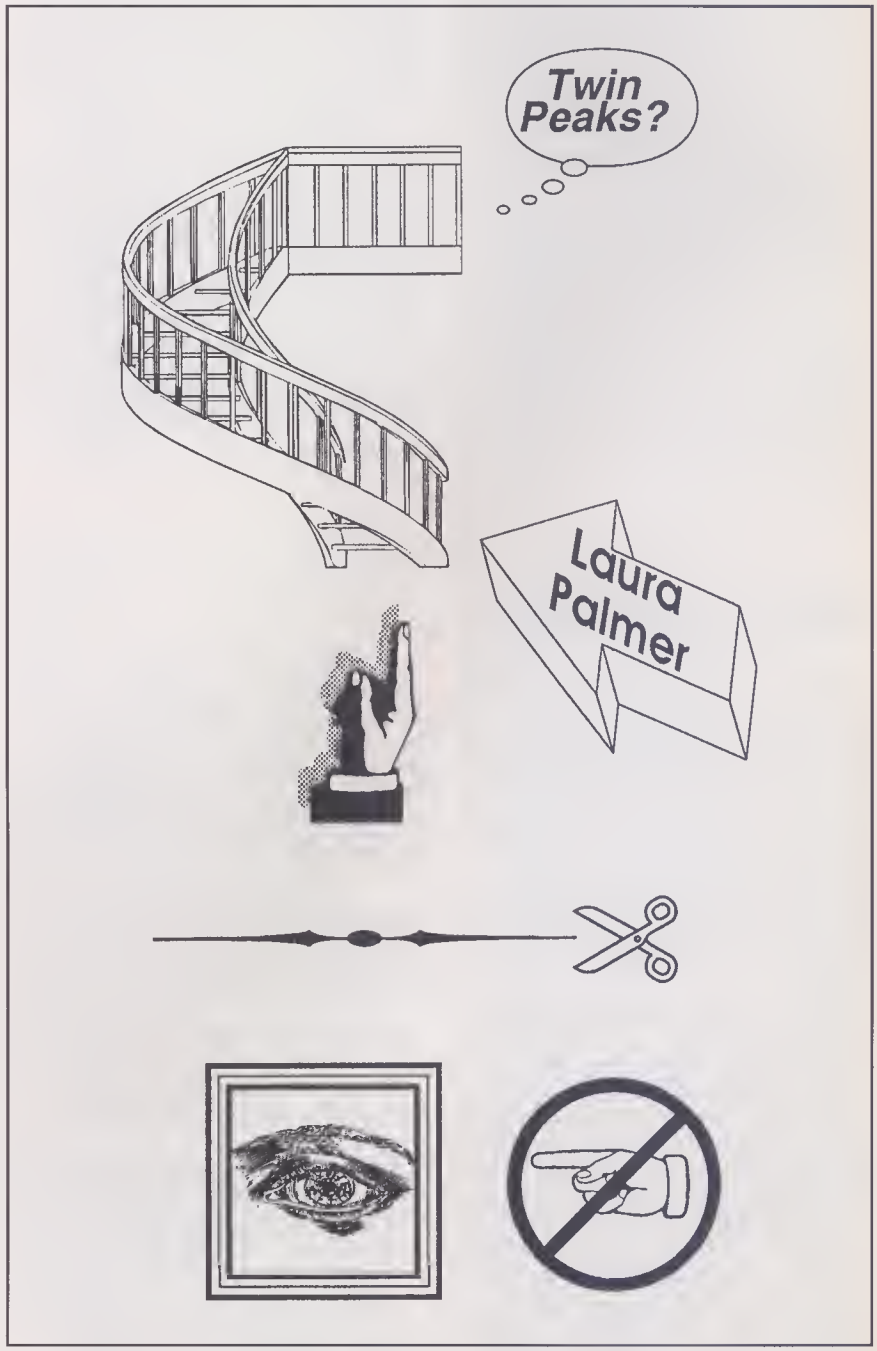


Figure 6. Streamline and Illustrator provide a variety of images.

with Illustrator as part of its package of symbols, borders, and letterforms. This package contains about 400 images that are ready to use. The images are grouped in documents, each containing 4-12 images. To use one of the images, just open the image document, select the image, copy it to the clipboard, and paste it into your document.

The frame around the eye is just one of several frames provided. I used it at its original size, scaled down the eye, and moved the eye into the center of the frame. The pointing hand is at its original size, but the "no" symbol was scaled to increase its size and then moved over the hand. If I had moved the hand over the circle instead, it would have hidden the crossbar (although I could have used Illustrator's "Paste in Back" command to move the hand behind the symbol).

The "no" symbol and the frame can be created easily in Illustrator. To make the frame, for example, you first select the rectangle tool. After positioning the mouse cursor at the upper right-hand corner and holding down the Shift key to constrain the rectangles to squares, you drag the rectangle to the size you want. (Holding down the Alt key would let you draw rectangles from the center out.) Then, scaling with the copy option, you create the thin, inner square. You can use the Transform Again command, Ctrl-D on the keyboard, to create the inner rectangle. You can set the line widths with the Paint menu.

As easy as it is to create this frame, it's already provided to save you time. Other symbols, such as the hands, the scissors, and the border can also be created in Illustrator, but it takes much more effort and talent. Fortunately, Illustrator gives you these, too.

The hand illustrates that you can create drop shadows with Illustrator. The staircase's thought balloon consists of one of the provided graphics plus text that I added using the text tool. This tool offers the 35 scalable fonts provided by the Apple Laserwriter. I chose the point size for the "Twin Peaks?" text to fit in the graphic.

The arrow with the "Laura Palmer" text was created in several steps. I first used the text tool to select the font, point size, and leading, and to type in the text. Then I used the arrow (one of the supplied graphics) and scaled it so that it would enclose the text. Then I used the shear tool to slant the arrow and text.

I'm not very good at drawing, but I can use The Complete Scanner to scan an image and then either convert it with Streamline or use it as a template in Illustrator. As a template, I can combine Illustrator's drawing tools along with its autotrace tool to create my own images. To help you use templates, Illustrator lets you view just the artwork, just the template, or the artwork and the template together.

These examples provide just a glimpse of Illustrator's power. If you need to include graphics in your documents and you can't draw, you need Adobe Illustrator and Streamline. If you can draw, you will be able to create great looking graphics easily. Adobe Systems is at 1585 Charleston Road, Mountain View, CA 94039, phone (800) 344-8335.

Illustrator: Reader Service 31
Streamline: Reader Service 32

Summary

You have many choices if you want Postscript output from a Laserjet printer. For an actual Laserjet printer, the Pacific Page cartridge is the best choice, with the AP-TeX fonts a good second choice for TeX users. For the rest of you, I don't recommend either of the software interpreters. If you need one, there are others on the market. I hope they are better.

Head of the class

Every now and then a product appears that is clearly far ahead of the competition. Quattro Pro from Borland is the latest entry in the spreadsheet arena and easily one of the best software products of any type I have seen. Quattro Pro is much more than an upgrade to Quattro. Although you can operate it with much the same look and feel of Quattro, the new version is so much better that it should be viewed as an entirely new product.

Quattro Pro is supplied on both 3.5- and 5.25-inch diskettes — a welcome feature if you have only one drive type and you fail to specify the distribution disk size. The effort to install the system is minimal. You can either use the automatic installation program or copy the files directly to your hard disk, in which case Quattro Pro will enter the installation program automatically the first time you try to run the main program. In either case, installation is easy and extremely intuitive. The complete package requires about 3 Mbytes of hard disk space.

If you later want to change any of the installation options, such as the printer type or the graphics card, you can do so directly through the menu system without having to reinstall the program. In fact, you can modify almost any feature of Quattro Pro in this way, including the system's entire look and feel.

Quattro Pro offers four separate display modes, which are selectable from the Options menu. Three of the modes are text based: 80×25-line mode, 80×43-line EGA mode, and 80×50-line VGA mode. The fourth display mode is graphics. Although all of the text modes are satisfactory, the graphics mode really shines. All modes provide full mouse support.

There are three distinct menu styles: the familiar Lotus 1-2-3 format, Quattro style (from the previous version), and a new style specific to Quattro Pro. The new menus are similar to the old version, so you won't have to learn an entirely new configuration, but they are different enough to offer distinct advantages. If you are moving to Quattro Pro from a system with 1-2-3 style menus, you will quickly get up to speed.

You can specify either tiled or stacked windows, and you can zoom tiled windows to full-screen size and back again with a single hotkey. A maximum of 32 windows are available. You can move between stacked windows by selecting the desired window from a menu list, or you can simply click on the corner of the window with the mouse. You can also use the mouse to move and size any window.

Quattro Pro lets you save a series of open files as a single named workspace. When you later reload the workspace, the previous window configuration is fully restored. If you often deal with groups of related spreadsheets, you will find this feature indispensable.

The program's most powerful feature is its ability to link data from several spreadsheets, which can eliminate the need to enter data more than once or to explicitly manipulate data from many spreadsheets to create summary information. Quattro Pro implements links as formulas that relate to a cell or a block of

data in any spreadsheet in your system. The link formula is automatically updated when the data in a referenced spreadsheet changes if that spreadsheet is open. If the spreadsheet is not open, you can explicitly update link references by choosing the Update Refs option from the menu. Also, when you load spreadsheets containing links, the program asks you if it should update any references to unloaded spreadsheets. The program will then load only the required data, rather than the entire referenced spreadsheet.

Quattro Pro offers outstanding graphics capabilities. You can choose any of 10 basic graphic types from a pictorial display including line graphs, pie charts, bar graphs, area graphs, and *x,y* graphs. A fast-graph feature is also available, in which you mark a block of data in the spreadsheet and let the program automatically partition the data, create the required labels and legends, and display the data in graphics form. Nine Bitstream Fontware typefaces are available for text in your graph: Swiss and Dutch in italic, bold, bold italic, and regular, and Courier in regular only. All the fonts can be scaled from 6 to 72 points.

One of Quattro Pro's easiest and most useful features is the graph annotator, which resembles many of the popular drawing packages and lets you put a variety of images into your graph, such as boxes, ellipses, and arrows. You can use the feature to control fonts in your graph and the placement of associated text, and you can change colors and patterns by selecting from a fixed pallet. A set of 35 Picture Pak clip art images is also included. I found that the annotator became an indispensable tool for creating sophisticated graphics in a very short period of time.

Quattro Pro's linking feature carries over into its graphing capabilities. The program lets you plot data from multiple spreadsheets in the same graph. In fact, none of the data in a graph need come from the active spreadsheet or from any loaded spreadsheet. This ability proves very useful when preparing summaries for quarterly and annual reports, obviating the need for a separate spreadsheet with only the summary data.

You can save any Quattro Pro graph as a named graph and recall it later. This lets you maintain a series of differently styled graphs representing related data. You can also prepare a slide show that presents a named graph for a fixed length of time before displaying the next named graph in the series or returning to your spreadsheet.

Quattro Pro lets you insert a graph directly into a spreadsheet as long as you're using the graphics display mode for the spreadsheet rather than one of the character modes. Once in the spreadsheet, the graph is automatically updated whenever any of its data changes. Although I liked this feature very much, I found that the graph's relatively small size often lowered its resolution to the point where it was no longer useful.

Printing spreadsheets can often be frustrating because of their awkward size. Quattro Pro helps alleviate this problem with a screen preview function that lets you see what your printed output will look like (except for minor font differences). The preview function provides both a ruler and three separate zoom levels. The ruler is most useful in determining the exact layout of your spreadsheet on a printed page, while the zoom feature provides greater detail for small sections of the page. You can display the preview in either portrait or landscape mode.

Quattro Pro easily imports spreadsheets created with either Quattro or Lotus 1-2-3 Version 2.01, and it can save them again in the same format. However, if you create any special graphics during your session, you must first save the spreadsheet in the new Quattro Pro format to avoid losing the information. You can then save the file in the older format. I frequently pass data among Quattro and Lotus users, and I found no problem with compatibility.

I have used Quattro Pro extensively as an accounting tool to track daily, monthly, and annual receipts and expenditures for a small business. The program's ability to link data from several spreadsheets has eliminated much of the duplication and additional work required to create summary spreadsheets. Its outstanding graphics capabilities, including the graphics link feature, have greatly helped my ability to spot trends and forecast performance.

Quattro Pro contains many features I have not mentioned, including context-sensitive help, a file manager, an interface to Paradox and dBase data files, outstanding macro facilities, and mathematical features such as regression analysis and linear programming. If you are looking for a spreadsheet package, I strongly recommend that you consider Quattro Pro.

Quattro Pro 1.0 requires 512 Kbytes of RAM, a hard disk, and DOS 2.0 or higher. If you want to use its full graphics capabilities, you will also need an EGA, VGA, CGA, AT&T, or Hercules graphics card or an IBM 8514 or 3270 PC. The retail price is $495. Quattro Pro is available from Borland International, 1800 Green Hills Road, P.O. Box 660001, Scotts Valley, CA 95066-0001, phone (408) 438-5300. — *Daniel McAuliffe*

Reader Service 33

Review notes

Microspell. If you do a lot of writing like I do, you need a spelling checker that is easy to use and doesn't get in the way. I've found one I think you will like.

Microspell, from Trigram Systems, is fast and full of features that make it a pleasure to use. It's system requirements are so modest that it will run on any IBM PC or clone with at least one floppy and 128 Kbytes of RAM.

I tried it on a 43,000-character LaTeX file. It quickly found the repeated words "the the" and some misspellings. When I ran the corrected file through it, Microspell took just over five seconds to check the entire document.

When Microspell finds a suspect word, it tells you what it thinks is wrong — for example, misspelling, irregular capitalization, or both. With a single keystroke you can tell Microspell to treat the word as correct for the rest of the document (or just for this instance).

The Save command lets you save a suspect word on an exception list and ignore it for the rest of the file. When you're done checking your document, you can add the exception list to an auxiliary dictionary, save it, or both.

Suspect words can be replaced by typing a replacement of up to 48 characters. Microspell then resumes checking from the start of the replacement to ensure that you didn't make a spelling error in the new text.

You can also replace suspect words by choosing a new word from a pop-up window containing a numbered list of up to nine replacements.

If you need to make major corrections around a suspect word, Microspell's Edit command displays the portion of the file immediately surrounding the word. Standard cursor movement keys let you move around the file, and standard editing keys let you delete characters and enter new text.

The lookup command lets you search Microspell's dictionary using * and ? wildcards. For example, "d*d*d" matches all words beginning with the letter "d" and containing at least two more "d's." What's especially nice is that you can also invoke Microspell from the DOS command line to look up individual words without performing a full spelling check. For example, "ms -w d*d*d" looks up words matching the pattern just described.

You can maintain Microspell's main dictionary by permanently adding and deleting words. There's also an auxiliary dictionary containing local additions to the dictionary that is loaded whenever Microspell is invoked.

The only problem I found with Microspell is that you can't have a local dictionary. Instead, you get an auxiliary dictionary for each file, which can be annoying if you are nesting files using an include-file capability. Trigram says this will probably be modified in the next release, which should be out by the time you read this.

You can also configure Microspell for your text processor. For example, you can have it ignore words that begin with command triggers. TeX users will find this especially useful.

If you aren't satisfied with your spelling checker, get Microspell. You won't be disappointed. The price is $69.95. Trigram Systems is at 5840 Northumberland St., Pittsburgh, PA 15217, phone (412) 422-8976. — *F. Pappas*

VP-Expert/SQL. This software from Gupta Technologies combines a rule-based expert system and the Structured Query Language for relational databases to help you create expert-system front ends for databases. Its potential applications span a wide variety of domains. The advantage of marrying VP-Expert to SQL is the latter's widespread use as a data-manipulation language for relational database systems and the potential to develop personal-computer expert systems that can access databases on larger systems.

The expert system uses knowledge-base rules to analyze data in a relational database. The rules have the general form "if C then B1 else B2," where condition C involves the fields of a table's current record and B1 and B2 are blocks of statements that affect variables defined in the knowledge base. VP-Expert/SQL's inference engine allows for automatic rule generation from induction tables, which contain in each row the value of the premises and of the conclusion (plus a confidence factor). Induction tables can be created either from SQL or directly via an editor that comes with the product.

VP-Expert/SQL automatically translates between its own language and SQL, eliminating the need to know SQL. The typical interrogation of database tables for expert system use does not require SQL's full power, so using the translation can be an advantage over using SQL directly. The product also allows dircet access to SQL.

VP-Expert/SQL's organization is helpful, and its nice graphic presentation of the menus lets the user consult, inspect, and modify the knowledge bases. There are minor discrepancies between the manuals and the real product, and not all of them are mentioned in the "read me" files, but they should not deter a seasoned programmer. The suggested retail price of $695 makes VP-Expert/SQL a good buy. Gupta Technologies is at 1040 Marsh Road, Suite 210, Menlo Park, CA 94025. — *Dan Simovici*

WSC'90

1990 Winter Simulation Conference

December 9-12, The Fairmont Hotel, New Orleans

Who Should Attend? The 1990 Winter Simulation Conference is an important event for everyone with interest in simulation. Practitioners applying simulation to manufacturing or service system analysis, researchers dealing with issues such as simulation language improvements and output analysis, and educators keeping up with simulation state-of-the-art will all find papers and presentations of interest.

Types of Presentations: The 1990 WSC focuses on discrete-event simulation and will feature basic tutorials, papers on applications and methodology, panel discussions, and state-of-the-art reviews. The keynote speaker is Philip Kiviat, originator of the simulation language GASP and leader in development of the language SIMSCRIPTII.

Software Tutorials: The 1990 WSC will feature an extensive schedule of tutorials on simulation software products presented by leading researchers and software developers. There are tutorials for all audiences: assuming no previous background in simulation, users who want to increase their knowledge of simulation, and researchers who want to learn about advanced and emerging topics.

Exhibit Area: There will be an extensive exhibit area for demonstrations of the latest advances in simulation software by software suppliers, with opportunities for you to speak to the software developers personally.

General Chair for WSC '90 is Dr. Randall P. Sadowski of Systems Modeling Corporation and Program Chair is Dr. Richard E. Nance of Virginia Tech.

For a copy of the preliminary program and/or for registration information contact:

1990 WSC Registration Headquarters
c/o EPIC Management, Inc.
8720 Red Oak Blvd.
Suite 224
Charlotte, NC 28217
1-800-447-6949 (available 9-5 Eastern time)

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

TRI-Ada
Soaring into the Nineties

The world's largest conference and exposition devoted exclusively to Ada development and trends

- Policy makers
- Engineering managers
- Research scientists
- Marketing & Sales executives
- MIS programmers, engineers and developers
- Automation experts
- Educators
- Technicians...

> ***"TRI-Ada...is one of the most important conferences in this nation."***
>
> Congressman John Murtha, Chair, House Subcommittee on Defense Appropriations.

...Whether you're an Ada novice or a seasoned pro, TRI-Ada '90 is for you. Expand your knowledge and expertise through:

- Wide-ranging technical and management sessions, tutorials and panel discussions led by top experts from industry, government and academia.
- A dynamic three-day exposition of the latest Ada tools and applications...Visit more than 100 exhibitors including Boeing, Contel, Harris Computer, Hewlett-Packard, IBM, McDonnell Douglas, Telesoft, Unisys, Westinghouse...and many more industry leaders.

- Exciting holiday entertainment and special events where you'll meet and mix with your peers...

For conference registration and exposition information, mail or fax the coupon to:

TRI-Ada '90
Danieli & O'Keefe Associates, Inc.
Conference Management
75 Union Avenue
Sudbury, MA 01776
FAX: 508-443-4715
Or call 1-800-848-5381
(In MA: 508-443-3330)

YES! ☐ **I am interested in attending TRI-Ada '90.** Send me conference program and registration information.

☐ **I am interested in exhibiting.** Send me exposition information.

Name: _____

Title: _____

Organization: _____

Address: _____

City, State, Zip: _____

Telephone: (_____) _____

TRI-Ada '90™

INDUSTRY • ACADEMIA • GOVERNMENT

December 3–7, 1990 • Baltimore Convention Center • Baltimore, MD

Sponsored by ACM/SIGAda and Baltimore SIGAda

NEW PRODUCTS

Contact or send releases to Steve Wilcox, *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; Compmail II, s.wilcox

Hand-held scanner puts text directly into applications

Caere Corp. says its Typist hand-held scanner can import text and numbers directly into a user's application at 500 words per minute.

The 300-dpi scanner was designed specifically for page recognition and includes the company's Anyfont optical character recognition system, which recognizes text regardless of the font or number of columns, according to the company. (In multicolumn formats, the scanner recognizes only the center column and throws away incomplete columns on either side.) Anyfont supports 11 languages and reads nonstylized fonts from 6-72 points in either landscape or portrait orientation.

Caere says the Typist offers a scanning speed of up to two inches per second and resets automatically to allow for either horizontal or vertical scanning. As a user scans text with the Typist, keyboard operation is interrupted, and data from the scanner enters the application via the keyboard buffer.

The Typist can also scan graphic images using image capture software supporting PCX, TIFF, and Pict output file formats. A dither pattern switch lets the user choose from four patterns when scanning photos or line art.

The Macintosh version requires 4 Mbytes of RAM and at least a Macintosh SE. It includes a SCSI interface box and costs $695. The 286 and 386 versions require a PC compatible with an AT or Micro Channel bus, 640 Kbytes of RAM, 2 Mbytes of expanded or extended memory, and a hard disk. They include an AT- or MCA-interface card and cost $595.

Reader Service 35

Fujitsu's VP2000 series expands

Fujitsu's VP2400/40 and VP2200/40 supercomputers feature four scalar processors and two vector processors, and offer maximum vector performance of 5 Gflops and 2 Gflops, respectively.

The systems are expected to be available in September of 1991. Monthly rental fees are about ¥79,000,000 for the VP2200/40 and about ¥103,000,000 for the VP2400/40.

Fujitsu has also announced its UXP/M operating system for the VP2000 series and M series (M-780/770/760 model groups). The operating system is based on Unix System V Release 4. Fujitsu expects to deliver UXP/M in April 1991, with a monthly rental fee for the basic software starting at ¥210,000 yen.

Reader Service 36

IBM System/390 encompasses new processors, architectures, operating systems

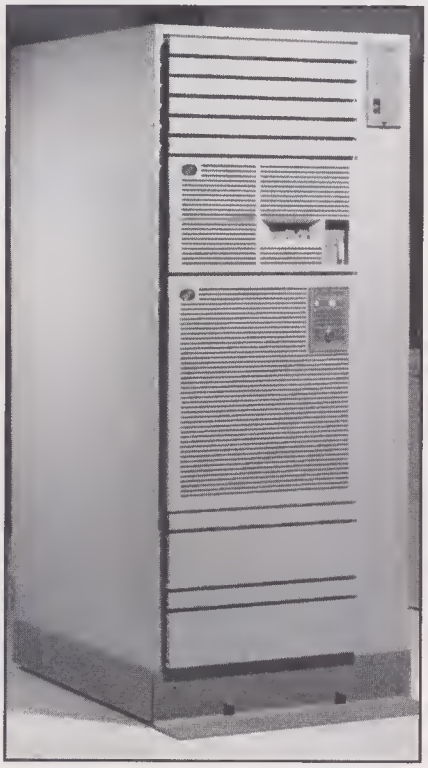
In a massive product announcement, IBM has rolled out its System/390, including new operating systems, connection architectures, and 18 new Enterprise System/9000 mainframe processors.

All ES/9000 processors support the new Enterprise Systems Connection (Escon) architecture, which uses fiber optic channels that transfer data at up to 10 Mbytes/sec. Escon lets users place I/O equipment as far as 5.6 miles (9 kilometers) from the processor, according to the company.

ES/9000 models also support logical partitioning between multiple operating systems at the same time to allow simultaneous use of production, development, maintenance, and test applications.

The 18 ES/9000 processors range from the rack-mounted, uniprocessor Model 120 with processor storage of up to 256 Mbytes, a maximum of 12 Escon and parallel channels, and no vector facilities, to the water-cooled, six-way multiprocessor Model 900 with up to 9 Gbytes of processor storage, a maximum of 256 Escon and parallel channels, and up to six vector facilities.

Fourteen of the new processors will support optional vector facilities, and some models allow overlapping of the scalar and vector execution elements and support multiple pipelines.



An air-cooled, rack-mounted ES/9000 model.

The ES/9000 series uses IBM's new Enterprise Systems Architecture/390. ESA/390 encompasses the VSE/ESA, VM/ESA, and MVS/ESA operating systems, which can also run programs and applications based on the System/370 architecture.

System/390 also includes integrated cryptography at central processor speeds and an external time-reference system that lets customers synchronize multiple processors and run them as one system.

Basic prices for the air-cooled processors range from about $70,500 to $3.12 million, with some models available now. Basic prices for the water-cooled models range from $2.45 million to $22.8 million, with most models available now.

IBM is offering direct upgrades for owners of ES/3090 J Models 180, 200, 280, 300, 400, 500 or 600, and ES/9370 Models 50, 60, 80, or 90. Also, entry-level ES/3090 uniprocessors can be upgraded to one of five ES/3090-9000T transition models.

Reader Service 37

Intergraph offers low-end workstations, servers

Intergraph has aimed its Series 2000 color graphics workstations and servers at such markets as mechanical design, mapping, and electronic publishing. Prices begin at $15,900.

Each workstation includes a 12.5-MIPS RISC processor, 160,000 2D vectors per second, and 100,000 3D vectors per second. The series includes such software as XNS, TCP/IP communications, Looking Glass (a graphical user interface), and Intergraph's implementation of Unix System V, Release 3.1 with Berkeley extensions.

The four workstation models are a desktop Interpro 2020 with a single 19-inch screen; the Interact 2020 with a single 19-inch screen, integral furniture, and a menu tablet, a desktop Interpro 2020 with dual 19-inch screens, and the Interact 2020 with dual 19-inch screens, integral furniture, and a digitizing table.

The Interserve 2000 includes a 12.5-MIPS RISC processor, 16 Mbytes of memory (expandable to 64 Mytes), a 200-Mbyte hard disk drive, a 1.44-Mbyte floppy disk drive, communications ports, and serial and parallel ports.

Reader Service 38

The Interact 2020 with dual 19-inch screens, integral furniture, and a digitizing table.

SCSI laser printer designed for graphics applications

NCR's new laser printer is targeted at graphics-intensive applications in the personal computer market. The SCSI interface of the 15-page/minute printer is designed to operate up to seven times faster than a standard parallel port and 100 times faster than a RS-232 serial port, according to the company.

The Model 6436-0301 allows shared-resource printing by accepting input from dual host systems. It also has a standard parallel interface, so that one host system can print on the SCSI port while another uses the parallel port. A mechanical output jogger separates jobs that print simultaneously.

The printer has standard emulations for the HP Laserjet Series II, HP 7475A, Diablo 630, Epson FX-86e/286e, IBM Proprinter XL, and line printers. It also includes 35 resident scalable fonts. Standard memory is 2 Mbytes and can be upgraded to 4 Mbytes. The retail price is $7,995.

Reader Service 39

Fujitsu mainframes embrace firm's Mission/DC

Five mainframes announced by Fujitsu are the first products in the company's "Multienvironment Information Systems Solution by Domain" Concept (Mission/DC), which divides mainframe functions into a database processing domain, an application processing domain, a communication processing domain, and a management processing domain.

The M-1800 mainframes range from the two-processor Model 20 to the eight-processor Model 85. The latter supports main storage of up to 2,048 Mbytes, system storage of up to 8,096 Mbytes, and up to 256 channels.

Monthly rental fees range from ¥78,000,000 for the Model 20 to ¥290,000,000 for the Model 85. The Models 20 and 30 are expected to be available by the end of April 1991, while the other models are due in the third quarter of 1991.

Reader Service 40

Ventura Publisher hits the Mac

Ventura Software has announced the Macintosh version of its previously DOS-only Ventura Publisher.

The new version offers such standard Ventura features as forward and backward automatic cross referencing, the ability to construct tables and scientific equations, long document capabilities (up to thousands of pages), letter kerning, and suppport for a wide variety of text and graphics files.

Features sepcific to the Macintosh version include a spelling checker, undo/redo, apply/cancel with dialog box chaining, movable dialog boxes, and on-line help menus.

The Macintosh edition requires at least a Macintosh Plus with 2 Mbytes of RAM, System 6.0.2, Finder 6.0.2, and a hard disk drive. The retail price is $795, with shipments scheduled for the fourth quarter of 1990.

Reader Service 41

IBM adds low-cost models to AS/400 line

IBM has added two low-cost computers to its AS/400 line and introduced a new AS/Entry model.

The AS/Entry line is based on IBM's Systcm/36 architecture. The Model Y10 has 1-2 Mbytes of main memory and 160-640 Mbytes of storage. It is priced at $11,000, plus $1,195 for version 6 of the System Support Program operating system.

The AS/400 Models CO4 and CO6 have 8 Mbytes of main memory and 640 Mbytes of storage. Users can increase the CO4's memory and storage capacity to 12 Mbytes and 960 Mbytes, respectively. Users can also upgrade both the CO4 and AS/Entry Y10 to a CO6 with up to 16 Mbytes of main memory and 1,280 Mbytes of storage.

The CO4 costs $14,500, plus $3,750 for version 3 of the OS/400 operating system. The CO6 is priced at $17,500, plus $6,350 for OS/400.

Reader Service 42

The IBM Application System/Entry Model Y10.

Hand-held scanner handles 256 gray levels

Logitech has announced the Scanman Model 256, a hand-held scanner featuring several proprietary analog and digital chips created to enhance hardware-to-software communication and more closely simulate how the human eye perceives the spectrum of gray scales.

The scanner includes Ansel image-editing software, designed to run under Microsoft Windows 3.0 and featuring 256 gray-level image manipulation. Ansel is compatible with such file formats as compressed and uncompressed TIFF, TIFF CCITT, BMP, PCX, and EPS. It accepts a wide variety of input, from simple line art to full 8-bit 256-gray-level information.

Ansel is designed to overcome some of Windows' limitations, according to Logitech. For example, Ansel takes over control of the standard VGA color palette from Windows to allow the display of the maximum number of gray levels.

Ansel's image-manipulation options include additional brightness, contrast, and gray-level adjustments. Users can resize, flip, rotate, and scale images, as well as smooth and sharpen textures and create negative and posterized images. The software also includes a "de-skew" command to realign documents that have been scanned at a angle.

The scanner has a retail price of $499 for the PC version ($599 for the Micro Channel version) and carries a lifetime hardware warranty.

Reader Service 45

Ada compiler designed for Wang VS systems

Wang VS Ada, an Ada compiler for Wang VS computer systems, conforms to the ANSI standard and has been validated by thc US Defense Dept.'s Ada Joint Program Office, according to Wang.

Wang VS Ada gives users access to Wang Integrated Image Systems and VS operating system services. It is integrated with the VS operating system environment and supports subroutine calls to other programming languages supported on the VS.

Prices for Wang VS Ada range from $5,000 for low-end systems to $60,000. The product is available immediately.

Reader Service 43

Wireless LAN achieves 2-Mbit transmission rate

NCR's Wave LAN is a wireless local area network that transmits data at 2 Mbits per second, according to the company. The Wave LAN is a printed-circuit board that fits a PC AT expansion slot and connects to an omnidirectional antenna.

NCR says a Wave LAN installation can include any number of PCs within a radius of 800 feet in an open office, with a working range from 250-1,000 feet, depending on the office environment. The company also claims that office partitions or normal office construction (except thick concrete or metal) will not block transmission.

The product is compatible with Novell Netware and has a suggested price of $1,390 for the network interface card, Novell drivers, and antenna. An optional Data Encryption Standard security feature is $90.

Reader Service 44

LAN Manager 2.0 features new pricing scheme

Microsoft's has announced a new pricing arrangement to support version 2.0 of its LAN Manager. LAN Manager 2.0's $995 price includes support for five users. User packs to support additional groups of up to 10 users are $995 each. An unlimited number of users can be supported for $5,495.

According to the company, Version 2.0 integrates with Microsoft Windows 3.0, supports 386 and 486 servers, and automatically configures to the server type, including installation of a 32-bit network-optimized I/O subsystem.

Reader Service 46

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

IEEE COMPUTER SOCIETY
Membership / Subscription Application

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

BENEFITS

Computer

You automatically receive *Computer* with membership. Written, reviewed, and refereed by experts, it features survey and tutorial articles covering the entire computer field, and departments such as new products, product reviews, standards, and a reader forum called "The Open Channel." (monthly).

Technical Committees

Participate in one or more of our 33 technical committees — networks of professionals with common interests in specialty areas within computer hardware, software, and applications.

Standards Working Groups

Participate in the development of the more than 100 standards projects currently sponsored by the society in such diverse areas as software engineering, local area networks, microprocessor buses, design automation, programming languages, and standards definitions.

Computer Society Press Books and Videos

Receive discounts of up to 50% on over 700 titles covering a broad spectrum of computer science topics such as networking, communications, advanced systems, image processing, security, artificial intelligence, and visualization. Over 120 new products are published annually.

Conferences and Tutorials

Choose from more than 100 conferences annually, ranging from large industry-oriented conferences replete with exhibits to small, highly interactive workshops. Members receive special low rates.

Schedule of Fees

To join: see item 1, 2, or 3.
To subscribe: see item 4.

Membership dues and periodical subscriptions are annualized to, and expire on, December 31. Pay full- or half-year rate depending on date of receipt by the Computer Society as indicated below.

	Half Year Mar 1-Aug 31	Full Year Sept 1-Feb 28
1 I don't belong to the IEEE and I want to join just the Computer Society	☐ $27.00	☐ $ 54.00
2 I don't belong to the IEEE and I want to join both the Computer Society and the IEEE*		
I reside in Region 1-6 (United States)	☐ $52.50	☐ $105.00
I reside in Region 7 (Canada)	☐ $48.50	☐ $ 97.00
I reside in Region 8 (Europe, Africa, or the Middle East)	☐ $48.00	☐ $ 96.00
I reside in Region 9 (Latin America)	☐ $44.50	☐ $ 89.00
I reside in Region 10 (Asia and Pacific)	☐ $43.50	☐ $ 87.00
3 I already belong to the IEEE and I want to join the Computer Society	☐ $11.00	☐ $ 22.00

*ACM members who join both IEEE and the Computer Society may deduct $5 off the full-year rate; $2.50 off the half-year rate.

IEEE Member Number _____

4 OPTIONAL PERIODICALS for new or current members

	issues per year		
IEEE Computer Graphics and Applications	6	☐ $12.00	☐ $ 24.00
IEEE Design and Test	4	☐ $11.00	☐ $ 22.00
IEEE Expert	6	☐ $10.00	☐ $ 20.00
IEEE Micro	6	☐ $10.50	☐ $ 21.00
IEEE Software	6	☐ $12.50	☐ $ 25.00
Transactions on:			
Computers	12	☐ $12.00	☐ $ 24.00
Knowledge and Data Engineering	4	☐ $ 7.50	☐ $ 15.00
Parallel and Distributed Systems	4	☐ $ 7.00	☐ $ 14.00
Pattern Analysis and Machine Intelligence	12	☐ $12.00	☐ $ 24.00
Software Engineering	12	☐ $11.00	☐ $ 22.00

Total amount remitted with this application $_____

DC residents add sales tax to optional periodicals.

☐ Checks accepted in Belgian, British, German, Swiss, Japanese, or US currencies. Checks must be drawn on a bank in the country of origin of the currency.

PRICES EXPIRE 12/31/91

☐ Visa ☐ Master Card ☐ American Express

Charge Card Number (Minimum Charge $10.) _____ Mo. Yr. Exp. Date

I hereby make application for Computer Society membership and, if elected, will be governed by IEEE's and the society's constitutions, bylaws, and statements of policies and procedures.

MAILING ADDRESS

Full signature _____ Date _____

☐ Male ☐ Female

First name _____ Middle initial(s) _____ Last name _____ Date of birth _____

Street address _____ City _____ State/Country _____ Zip _____

EDUCATION (highest level completed) _____ Course _____ Degrees received _____ Date _____

Name of educational institution _____

OCCUPATION

Title or Position _____

Firm name _____

Firm address _____

City _____ State/Country _____ Zip _____

ENDORSER (an IEEE member, Senior Member, or Fellow who knows you professionally)

Endorser's signature _____

Name (print in full) _____ IEEE Member No. _____

Street address _____

City _____ State/Country _____ Zip _____

Return to: IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720–1264 USA.
Residents of Europe mail to: IEEE Computer Society, 13, Avenue de l'Aquilon, B-1200, Brussels, BELGIUM.
Asian/Pacific residents mail to: IEEE Computer Society, Ooshima Building, 2-19-1 Minami-Aoyama, Minato-ku, Tokyo 107 JAPAN.

PC1090

ADVERTISER INDEX

Academic Press, Inc. 119
ACM SIGSOFT '90 78
CACI Products Company 1
Cadre Technology Inc. 4-5
Cap Gemini Publishing 74
Data Compression Conference 99
Design Automation Conference 22
Elsevier Science Publishing Co. 64
Hecht-Nelson Neurocomputers 20-21
Hyperception, Inc. . . . . Cover IV
ICCD '91 108
IEEE Computer Society Membership 95
IEEE Computer Society Ombudsman 77
IEEE Educational Activities 56
Journal of Information Processing 72
King Fahd University 32, 71
National University of Singapore 75, 115
Oakland Group, Inc. 6
OOPSLA '91 . . . . . Cover III
Physical Design Workshop 105
RTSS '90 81
StarSys, Inc. 43
Telesoft . . . . . Cover II
TRI-Ada '90 91
VLDB '91 44
WSC '90 90
Classified Advertising 109-115

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northern California and Pacific Northwest: John D. Vance & Associates, Inc., 4030 Moorpark Ave., Suite 116, San Jose, CA 95117; (408) 741-0354.

Southern California and Mountain States: Richard C. Faust Co., 24050 Madison St., Suite 100, Torrance, CA 90505; (213) 373-9604.

Midwest: The Kingwill Company, 4433 W. Touhy Ave., #540, Lincolnwood, IL 60646; (708) 675-5755.

East Coast: Atlantic Representative Group, 349 Maple Pl., Keyport, NJ 07735; (908) 739-1444.

New England: Arpin Associates, P.O. Box 6444, Holliston, MA 01746; (508) 429-8907.

Europe: Heinz J. Görgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; fax: (0 21 53) 8 99 89.

Southwest/Southeast: Heidi Rex, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

COMPUTER, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

PRODUCT INDEX

	RS#	PG#
3Com	146	98
Academic Press, Inc.	8	119
Accutek Microcircuit	120	97
Adobe Systems	31-32	88
Analog Devices	121, 122	97
Arbortext	22	83
Arcom	136	98
Ariel	135	98
Borland International	33, 137	89, 98
CACI Products Company	—	1
Cadre Technology Inc.	2	4-5
Caere Corp.	35	92
Cap Gemini Publishing	7	74
CMS Enhancements	138	98
Corollary	139	98
Custom Applications	24	84
Data Translation	140	98
Dynatech	141	98
Elsevier Science Publishing Co.	6	64
Fujitsu	36, 40	92, 93
Gould AMI	123	97
Grid Systems	142	98
Hecht-Nelson Neurocomputers	4	20-21
Hewlett-Packard	143	98
Hyperception, Inc.	9	C.IV
IBM	37, 42	92, 94
Inset Systems	26	84
Intergated Device	124, 125	97
Intergraph	38	93
Intermetrics	126	97
Kinch Computer	23	83
Lasergo	25	84
Logitech	45	94
Microsoft	46	94
Motorola	127	97
NCR	39, 44	93, 94
Oakland Group, Inc.	3	3
Pacific Data Products	27-29, 144	85, 98
Personal TeX	21	82
Piiceon	145	98
RO Associates	128	97
Sipex	129	97
SquareD	130	97
StarSys, Inc.	5	43
Telesoft	1	C.II
The Complete PC	30	85
Ventura Software	41	93
Wang	43	94
Xicor	131	97

COMPUTER
Reader Service Card

October 1990 issue (Card void after April 1991)
Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

201 Publications catalog	**203** Student membership application
202 Membership information	**204** Application for senior membership

Reader interest
(Circle what you liked; add comments on the back)

A	Dependability Modeling
B	Fault Tolerance
C	Clock Synchronization
D	Built-In Self Testing
E	Chain Reactions
F	Taulbee Survey
G	Standards
H	CS News
J	Update
K	Product Reviews
L	New Products
M	IC Announcements
N	Micro Announcements
P	Call for Papers
Q	Calendar
R	Book Reviews
S	CS Mags

Product information
(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

COMPUTER
Reader Service Card

October 1990 issue (Card void after April 1991)
Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

201 Publications catalog	**203** Student membership application
202 Membership information	**204** Application for senior membership

Reader interest
(Circle what you liked; add comments on the back)

A	Dependability Modeling
B	Fault Tolerance
C	Clock Synchronization
D	Built-In Self Testing
E	Chain Reactions
F	Taulbee Survey
G	Standards
H	CS News
J	Update
K	Product Reviews
L	New Products
M	IC Announcements
N	Micro Announcements
P	Call for Papers
Q	Calendar
R	Book Reviews
S	CS Mags

Product information
(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

COMPUTER
Reader Service Card

October 1990 issue (Card void after April 1991)
Print address below.

Name _____

Title _____

Company _____

Address _____

City _____

State/ZIP _____

Country _____ Phone _____

Please send (Circle those you want)

201 Publications catalog	**203** Student membership application
202 Membership information	**204** Application for senior membership

Reader interest
(Circle what you liked; add comments on the back)

A	Dependability Modeling
B	Fault Tolerance
C	Clock Synchronization
D	Built-In Self Testing
E	Chain Reactions
F	Taulbee Survey
G	Standards
H	CS News
J	Update
K	Product Reviews
L	New Products
M	IC Announcements
N	Micro Announcements
P	Call for Papers
Q	Calendar
R	Book Reviews
S	CS Mags

Product information
(Circle numbers for products and advertisements on which you want more information)

1	21	41	61	81	101	121	141	161	181
2	22	42	62	82	102	122	142	162	182
3	23	43	63	83	103	123	143	163	183
4	24	44	64	84	104	124	144	164	184
5	25	45	65	85	105	125	145	165	185
6	26	46	66	86	106	126	146	166	186
7	27	47	67	87	107	127	147	167	187
8	28	48	68	88	108	128	148	168	188
9	29	49	69	89	109	129	149	169	189
10	30	50	70	90	110	130	150	170	190
11	31	51	71	91	111	131	151	171	191
12	32	52	72	92	112	132	152	172	192
13	33	53	73	93	113	133	153	173	193
14	34	54	74	94	114	134	154	174	194
15	35	55	75	95	115	135	155	175	195
16	36	56	76	96	116	136	156	176	196
17	37	57	77	97	117	137	157	177	197
18	38	58	78	98	118	138	158	178	198
19	39	59	79	99	119	139	159	179	199
20	40	60	80	100	120	140	160	180	200

Editorial comments

I liked: _____

_____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

_____

For reader-service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

COMPUTER
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

Editorial comments

I liked: _____

_____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

_____

For reader-service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

COMPUTER
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

Editorial comments

I liked: _____

_____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

_____

For reader-service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader-service cards only.

COMPUTER
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

IC Announcements

Company, Model, Function	Comments	R.S. No.
Accutek Microcircuit Accutek Forty Plus High-density memory modules	One-Mword × 8-bit CMOS DRAM modules for 40 MHz computers come in 1M and 4M versions, operate in read-write and read-write-modify modes for high-speed applications, and provide data-in, data-out, and write-enable lines to separate pins on separate buses. JEDEC 64-pin socket with 50-mil pin-to-pin spacing; 3.85″ wide × 0.90″ high. Cost: from $89.	120
Analog Devices AD9014 ADC	ECL-compatible 14-bit ADC is factory tested at 10 megasamples/s with analog input frequencies of 540 kHz and 2.3 MHz. Minimum spurious-free dynamic range is 90 dB, decreasing to 86 dB with input frequencies up to 4.3 MHz. Features minimum 75-dB signal-to-noise ratio and 90-dBc two-tone intermodulation distortion. Cost (100s): from $2,800.	121
Analog Devices ADV7141, 46, 48 Video RAM-DAC IC	Pin-compatible upgrade for RAM-DAC IC on standard VGA systems implements Edsun's Continuous Edge Graphics algorithm. Eliminates aliasing, provides realistic colors and subtle shadings, and displays text with laser-printer quality. Available in 44-pin PLCC 35-, 50-, and 66-MHz clock versions and 28-pin plastic DIP. Cost (1,000s): $30-$50.	122
Gould AMI S63 Series Military CMOS ROMs	High-speed CMOS ROMs, including versions with 16-Kbit to 4-Mbit capacity, comply with Mil-Std-883C. The memories provide access times up to 100 ns and require a single +5V power supply. The 4-Mbit ROM will be available in 512K × 8 and 256K × 16 configurations. Packages include 24- and 28-pin ceramic DIP and 32-pin CLCC. Cost (1,000s): $30-$105.	123
Integrated Device Technology IDT10494, 100494, 101494 ECL SRAMs	64K static RAMs use IDT's 0.6 (Leff) micron BiCEMOS II process, which typically limits power consumption to 700 mW. Suited for implementing caches in ECL RISC systems, the devices are organized as 16K × 4 bits with separate data inputs and outputs and are available with access times of 7, 8, 10, and 15 ns. Cost (100s) of 7 ns DIP version: $125.	124
Integrated Device Technology IDT7RS355 Floating-point library	Floating-point library for R3000 RISC systems eliminates the need for an FPA chip, thus saving space on the board and allowing the development of applications that need single- and double-precision floating-point capabilities but not the high performance of the coprocessor. Cost: $1,295.	125
Intermetrics C cross compilers and debuggers	Whitesmiths' optimizing C cross compilers and CXDB C source-level cross debuggers for the Zilog Z80, Motorola 6809, Hitachi 64180 microcontrollers are preintegrated and provide a C development environment for embedded systems designers. Cost: compiler packages hosted on IBM PC, from $1,800; CXDB on IBM PC, $1,500.	126
Motorola 68HC11K Series 8-bit microcontrollers	Two additions to Motorola's 8-bit 68HC11 microcontroller family double the bus speed of the original family members. The K4 and 711K4 feature 768 bytes of RAM, 640 bytes of EEPROM, 62 I/O lines, a 16-bit timer, four 8-bit pulse-width modulators, 8-channel 8-bit A/D, and enhanced serial peripheral and communication interfaces. No price given.	127
RO Associates RO48-D512 DC-DC converter	Dual-output DC-DC converter produces 5V at 10A and 12V at 8A with a total maximum power of 125 watts. Each output can be either a plus or minus 5V or 12V; input range is 36-66 VDC with input overvoltage protection up to 100V. Module requires no heat sink or fan and complies with Mil 810D, UL, CSA, and IEC requirements. Cost (singles): $325.	128
Sipex SP9462 Data acquisition system	Multichannel DAS in a 68-pin grid array features an 8-differential or 16-single-ended input multiplexer, programmable gain instrumentation amplifier and sample-and-hold, and a 12-bit A/D converter. All timing, control, and voltage references are internal to the package, making the DAS a complete subsystem. Cost (100s): $100.	129
Square D Micro-1 Microcontroller	Controller's internal logic equals 160 relays and holds 600 steps of EEPROM memory. Controller has eight inputs and six outputs (expandable to 16 and 12), 80 timers, and 45 counters. Programming is via a small, plug-in program loader. Up to 32 controllers can be linked to a PC to monitor remote control panels and machines. Cost: $200.	130
Xicor X24C02 EEPROM	CMOS 2-Kbit serial EEPROM is internally organized as one 256 × 8 page. It features a serial interface and software protocol that allows operation on a single two-wire bus. It is available in commercial and industrial temperature ranges and in 8-pin plastic DIP and plastic SOIC. Cost (100s): From $1.45 to $2.21, depending on temperature ranges and packaging.	131

Microsystem Announcements

Company, Model, Function	Comments	R.S. No.
Ariel Quintprocessor Next DSP board	This digital signal processing board for the Nextbus features five Motorola 27-MHz DSP56001 DSPs, four as slave processors and one as an I/O processor that manages DRAM memory, SCSI mass storage, and interprocessor communication. The board is compatible with Ariel's BUG-56, a symbolic debugger bundled with the Next computer. Cost: $6,995.	135
Arcom VFDSC SCSI/disk controller	A VMEbus board with a SCSI controller port for up to eight devices and an independent floppy disk port for up to four drives. Compatible with STEbus OS-9 driver software. LEDs display the status of the SCSI lines. The disk port can be polled or interrupt driven. Cost: £485.	136
Borland Paradox 3.5/ Paradox SQL Link	The new version of this relational database management system works with Paradox SQL Link to give users access to remote SQL data. Paradox includes new memory management technology, and Paradox SQL Link provides automatic configuration and transparent connection to SQL servers. Cost: Paradox 3.5, $795; Paradox SQL Link, $495; Paradox Multipack (five users) $995.	137
CMS Enhancements PS/1 memory modules	Two single-slot memory modules that provide up to 512 Kbytes or 2 Mbytes of expanded memory for the IBM PS/1 computer. Cost: 512 Kbytes, $169; 2 Mbytes, $595.	138
Corollary 8X2 board Multiport system	A multiport subsystem providing RS-232 serial connections for 8- and 16-user PC AT 386- and 486-compatible systems. One board connected to two 8-port 8/tc+ terminal concentrators supports up to 16 users; four boards support as many as 64 users. Each 8-port concentrator can be up to 1,000 feet from the host computer Cost: $995 for a basic kit, including an 8X2 board, an 8/tc+ terminal concentrator, RS-422 host cable, RS-232 cable, documentation, and drivers	139
Data Translation DT2812 I/O board	The DT2812 DMA analog I/O board and DT2812-A digital I/O board fit the long or short slots of PC XT and AT machines. Designed for the OEM market, the boards include counter/timers and software (MS-DOS device driver, toolkit subroutine library, and DT/Gallery). Cost: DT2812, $629; DT2812-A, $695.	140
Dynatech 386 SBC/486 SBC Single-board computers	Two semicustom, single-board VMEbus computers based on the 80386 and 80486. Available with 4, 8, or 16 Mbytes of RAM and 32 Kbytes of cache memory. They can run DOS or Unix at 25 MHz or 33 MHz, and they include two RS-232 and six RS-422 serial ports, one parallel port, floppy disk interface, hard disk interface, keyboard interface, clock, and VGA graphics. Cost: From $7,500 for the 386 SBC.	141
Grid Systems Gridcase 1550sx Laptop computer	The 12-pound battery-powered Gridcase 1550sx features a built-in pointing device, 20-MHz 386SX processor, 60-Mbyte hard drive, 640X480 VGA backlit LCD screen, 2 Mbytes of RAM. Battery life is up to three hours, and it recharges in about two hours. Cost: $6,295; with optional 120-Mbyte hard drive, $6,995.	142
Hewlett-Packard Deskjet 500 Printer	A 300-dpi, plain-paper printer compatible with Windows 3.0 (An enhanced HP driver that allows scaling fonts under Windows 3.0 is scheduled to ship later this year). Prints up to three pages of text or two pages of graphics per minute. Cost: $729. HP Deskjet PLUS and Deskjet owners can upgrade for $175 and $225, respectively.	143
Pacific Data Products Pacific T Memory Printer memory board	An memory board allowing users to expand the standard 512-Kbyte memory of TEC engine (Epson, Toshiba, AT&T, MCR, Mannesmann Tally, and Packard Bell) laser printers. The board can be upgraded from 2 to 4 Mbytes by adding memory chips. Cost: 2 Mbytes, $399; 4 Mbytes, $699.	144
Piiceon 4-Mbyte board for IBM 80 A-21	A 4-Mbyte memory board that plugs into the motherboard of the IBM 80 A-21 (a 386-based upgrade of the PS/2 80 that comes with 4 Mbytes standard). Cost: $1,895.	145
3Com 3Server/600 486-based server	A server based on a 33-MHz 486 chip, otherwise identical to 3Com's 386-based 3Server/500 with the 486/33 Option installed. The 486/33 Option is also available separately for $6,895. The price for the 3Server/600's basic configuration is $24,250.	146

CALL FOR PAPERS

Data Compression Conference

(Sponsored by the IEEE Computer Society TCC in Cooperation with NASA/CESDIS)

Snowbird, Utah
April 8 - 10, 1991

Program Committee:
A. Blumer, *Tufts U.*
R. Capocelli, *U. Rome*
J. Cleary, *U. Calgary*
P. Elias, *MIT*
I. Daubechies, *Bell Labs*
R. Gray, *Stanford U.*
D. Hirschberg, *UC Irvine*
A. Lempel, *Technion*
V. Miller, *IBM*
J. Reif, *Duke U.*
D. Sheinwald, *IBM*
J. Storer, *Brandeis U.*
J. Tilton, *NASA*
J. Vitter, *Brown U.*
A. Wyner, *Bell Labs*
J. Ziv, *Technion*

Co-General Chairs:
J. Reif, *Duke U.*
J. Storer, *Brandeis U.*

Publicity Chair:
R. Miller, *NASA/CESDIS*

Registration Chair:
M. Cohn, *Brandeis U.*

THEME: Data compression has become an essential component of high speed communications and mass storage. This conference will provide an international forum for current work on data compression and related areas. Topics of interest include but are not limited to: *Coding theory, quantization theory, parallel algorithms and compression hardware, algorithms for specific types of data (including text, images, video, speech, maps, instrument data, graphics, animation, and bit-maps), transform methods, wavelet and fractal techniques, string searching and manipulation, closest-match retrieval, theory of minimal length encoding and applications to learning, system issues (including error control, data security, and indexing), medical imagery, scientific and space data archives, data compression standards.*

SUBMISSION: Authors are requested to submit 3 copies of an extended abstract of at most 10 pages by **December 14, 1990** to:

Prof. Martin Cohn
Computer Science Dept.
Brandeis University
Waltham, MA 02254
(fax: 617-736-2741, email: marty@cs.brandeis.edu)

NOTIFICATION: Authors will be notified of acceptance or rejection by a letter mailed on 1/15/91. A final draft is due 2/15/91. Students presenting papers are eligible for financial support to attend the conference.

ADVANCE REGISTRATION: Registration includes reception, banquet, coffee breaks, and proceedings. For registrations *received before 3/1/91*, the fee is $295.00 for IEEE or affiliate members or $370.00 for non-members; there is an additional $100 late fee for registrations after 3/1/91.

HOTEL: Space is limited; rooms may NOT be available after 3/1/91. Room rates are $42 per person for a double room and $84 for a single room. Call the *Cliff Lodge, Snowbird, UT 84092*, 800-453-3000 or 801-742-2222.

TRANSPORTATION: The hotel may be reached by limousine or public bus from the Salt Lake City airport.

REGISTRATION FORM

NAME: _____

AFFILIATION: _____

ADDRESS: _____

_____

_____

PHONE: _____ EMAIL: _____

IEEE member ☐ yes ☐ no AMOUNT ENCLOSED: _____

MAIL TO:

DCC '91
Att. Myrna Fox
Computer Science Dept.
Brandeis University
Waltham, MA 02254

CALL FOR PAPERS

Conf. on Advanced Research in VLSI: Mar. 25-27, 1991, Santa Cruz, Calif. Submit five copies of draft paper by **Oct. 15, 1990,** to Carlo H. Sequin, Univ. of California, CS Div., 529B Evans Hall, Berkeley, CA 94720.

ICDCS 91, 11th Int'l Conf. on Distributed Computing Systems: May 20-24, 1991, Arlington, Texas. Submit five copies of abstract and paper by **Oct. 23, 1990,** to Benjamin W. Wah, ICDCS 91, Coordinated Science Lab, MC228, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801-3082, phone (217) 333-3516, fax (217) 244-1764, e-mail wah%aquinas@uxc.cso.uiuc.edu.

Int'l J. of Computer Simulation plans a special issue in early 1991 on simulation of multiple processor/distributed systems. Publisher: Ablex. Submit four copies of complete manuscript by **Oct. 30, 1990,** to Kallol Bagchi, Inst. of Electronic Systems, Aalborg Univ., Fredrik Bajers Vej 7, DK-9220 Aalborg 0, Denmark, fax (45) 98-156-740, e-mail kkb@iesd.auc.dk.

Second Int'l Conf. on Local Communication Systems: June 26-28, 1991, Palma, Spain. Sponsor: Int'l Federation for Information Processing. Submit six copies of abstract/paper by **Oct. 31, 1990,** to Ramon Puigjaner,

Call for papers and referees for *Computer*

Computer seeks articles for inclusion in upcoming special issues.

Distributed Computing Systems has been selected as the theme for the **August 1991** issue. Prospective authors are invited to submit tutorial, survey, descriptive, case-study, application-oriented, or pedagogic manuscripts. See the July 1990 issue of *Computer* (p. 120) for complete information.

Abstracts are due by **November 15, 1990**, and the deadline for full manuscripts is **January 1, 1991**. Notification of decisions is set no later than **March 15, 1991**, and the final version of each manuscript is due no later than **May 1, 1991**.

Submittals and questions should be directed to either of the guest editors, Mukesh Singhal, Dept. of Computer and Information Science, Ohio State University, Columbus, OH 43210, phone (614) 292-5839, e-mail singhal@cis.ohio-state.edu; or Thomas L. Casavant, Dept. of Electrical and Computer Eng., University of Iowa, Iowa City, IA 52242, phone (319) 335-5953, e-mail tomc@eng.uiowa.edu.

For submittal to Computer, *manuscripts must not have been previously published or currently submitted for publication elsewhere. Each manuscript should be no more than 32 typewritten, double-spaced pages long, including all text, figures, and references. Each submittal should include a cover page that contains the title of the article, the full name(s) and affiliation(s) of the author(s), complete postal and electronic address(es) of all the authors as well as their telephone and fax number(s), a 300-word abstract, and a list of keywords identifying the central issues of the manuscript's contents. The final manuscript should be approximately 8,000 words in length and contain no more than 12 references.*

If you are willing to review articles for any of these special issues, please send a note listing your research interests to Bruce Shriver, editor-in-chief of Computer, *or to one of the guest editors listed for the particular issue. Shriver may be reached at the University of Southwestern Louisiana, PO Drawer 42730, Lafayette, LA 70504-2730, phone (318) 231-5811, fax (318) 265-5472, e-mail b.shriver@compmail.com or, on Internet, shriver@usl.edu.*

Multimedia Information Systems will be the theme of the **October 1991** issue. Manuscripts reporting survey, original research, design and development, and applications of multimedia technology are sought immediately in the following areas:

- A tutorial/survey on multimedia information technology
- Authoring techniques for multimedia systems
- Storage and indexing of multimedia data
- Communications of multimedia data
- Dynamic data in multimedia systems
- User interfaces for multimedia systems
- Browsing in multimedia systems
- Special workstations for multimedia applications
- Experiences with implementing multimedia applications
- Movies on a computer
- Intelligent television technology
- Interactivity and multimedia information systems
- Combining video with 3D

An abstract of each manuscript is due no later than **December 1, 1990**, and eight copies of each full manuscript must be submitted by **February 1, 1991**. Notification of decision is set **May 1, 1991**, and the final version of each manuscript must be submitted by **July 1, 1991**.

Submissions and questions should be directed to A. Desai Narasimhalu, Inst. of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Singapore 0511, phone (65) 772-2002, fax (65) 778-2571, e-mail issad@nusvm; or Stavros Christodoulakis, Department of Computer Science, Davis Building, University of Waterloo, Waterloo, Ont., Canada, phone (519) 888-4452 or (30) 821-64846, fax (519) 885-1208 or (30) 821-53571, e-mail schristodoul@waterloo.edu.

Heterogeneous Distributed Database Systems is the theme planned for the **December 1991** issue. See the August 1990 issue of *Computer* (p. 115) for complete information.

Abstracts of the manuscripts are due no later than **January 1, 1991**, and eight copies of the full manuscripts must be submitted by **April 1, 1991**. Notification of decisions is **July 1, 1991**, and the final version of each manuscript is due **September 1, 1991**.

Submissions and questions should be directed to Sudha Ram, Department of Management Information Systems, Eller School of Management, University of Arizona, Tucson, AZ 85721, phone (602) 621-2748, e-mail ram@mis.arizona.edu on Internet or ram@arizmis on Bitnet.

Univ. de les Illes Balears, Carretera de Valldemossa, km. 7.6, 07071 Palma, Spain, phone 34 (71) 207111, fax 34 (71) 758061 or 200741, e-mail dmilan91@ps.uib.es.

Int'l Symp. on Software Reliability Eng.: May 17-18, 1991, Austin, Texas. Cosponsors: IEEE Computer Society Technical Committee on Software Eng. and the Nat'l Aeronautics and Space Administration. Submit five copies of full paper by **Nov. 1, 1990,** to John C. Munson, Computer Science Div., Univ. of West Florida, Pensacola, FL 32514, phone (904) 474-2989, e-mail jmunson@dcsnet.uwf.edu.

Computer Graphics and Education 91: Apr. 4-6, 1991, Barcelona, Spain. Sponsor: Int'l Federation for Information Processing. Submit six copies of extended abstract or draft paper by **Nov. 1, 1990,** to Steve Cunningham, Computer Science Dept, California State Univ. at Stanislaus, Turlock, CA 95380, phone (209) 667-3176, e-mail rsc@altair.csustan.edu; or Roger Hubbold, Computer Science Dept., Univ. of Manchester, Oxford Road, Manchester M13 9PL, UK, phone (44) 61-275-6158, e-mail hubbold@uk.ac.man.cs.

Int'l J. Computer-Aided VLSI Design plans a special issue on VLSI implementations of artificial neural networks. Publisher: Ablex. Submit five copies of full papers by **Nov. 1, 1990,** to Medhi Zargham, Computer Science Dept., Southern Illinois Univ., Carbondale, IL 62901-45411, phone (618) 453-6048, e-mail ga3633@siucvmb on Bitnet.

21st Int'l Symp. on Multiple-Valued Logic: May 26-29, 1991, Victoria, Canada. Submit five copies of manuscript and abstract by **Nov. 1, 1990,** to Wayne Current, Electrical Eng. and Computer Science Dept., Univ. of California at Davis, Davis, CA 95616, phone (916) 752-1839 or 0583 (for the Americas); Michel Israel, IIE-CNAM, 18, Allee Jean Rostand, BP 77, 91002 Evry Cedex, France, phone 33 (1) 60-77-97-40 (for Europe and Africa); or Okihiko Ishizuka, Electronic Eng. Dept., Miyazaki Univ., Miyazaki-Shi 889-21, Japan, phone (81) 985-58-2811 (for Asia and the Pacific).

Eighth Int'l Conf. on Testing Computer Software: June 17-21, 1991, Washington, DC. Sponsor: Data Processing Management Assoc. Educational Foundation. Submit four copies of two-page abstract by **Nov. 1, 1990,** to Genevieve Houston-Ludlam, Frontier Technologies, 190 Admiral Cochran Dr., Suite 180, Annapolis, MD 21401, phone (301) 266-8244, fax (301) 224-3840.

North Am. Fuzzy Information Processing Soc. Workshop: May 15-17, 1991, Columbia, Mo. Submit one-page abstract by **Nov. 1, 1990,** to Jim Keller, Electrical and Computer Eng., Univ. of Missouri — Columbia, Columbia, MO 65211, phone (314) 882-7339, fax (314) 882-0397.

Fifth Int'l Conf. on Logic Programming: June 25-28, 1991, Paris. Cosponsors: Assoc. of Logic Programming et al. Submit six copies of paper by **Nov. 3, 1990,** to Koichi Furukawa, ICOT, Mita Kokusai Bldg., 21 F 4, 28 Mita 1, Chome, Minato-ku, Tokyo, Japan 108, e-mail furukawa@icot.or.jp.

DAC 91, 1991 IEEE/ACM Design Automation Conference: June 16-21, 1991, San Francisco. Submit eight copies of paper and 60-word abstract by **Nov. 7, 1990,** to Alfred E. Dunlop, c/o MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4333.

CICC 91, IEEE Custom Integrated Circuits Conf.: May 12-15, 1991, San Diego, Calif. Submit summary and abstract by **Nov. 8, 1990,** to Roberta Kasper, CICC 91, 1597 Ridge Rd. W, Suite 101C, Rochester, NY 14615, phone (716) 865-7164.

CVPR 91, IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition: June 3-7, 1991, Lahaina, Maui, Hawaii. Submit four copies of complete paper by **Nov. 12, 1990,** to Gerard Medioni, Inst. for Robotics and Intelligent Systems, PHE 204, mc 0273, Univ. of Southern California, Los Angeles, CA 90089-0273, e-mail medioni@dworkin.usc.edu.

SAC 91, 1991 Symp. on Applied Computing: Apr. 3-5, 1991, Kansas City, Mo. Sponsor: Univ. of Missouri — Kansas City. Submit five copies of paper by **Nov. 13, 1990,** to Vijay Kumar, SAC 91, Computer Science Telecommunications Program, Univ. of Missouri — Kansas City, 5100 Rockhill Rd., Kansas City, MO 64110-2499.

ICCI 91, Int'l Conf. on Computing and Information: May 27-29, 1991, Ottawa, Canada. Sponsors: Carleton Univ, Ottawa; Natural Sciences and Eng. Research Council of Canada. Submit six copies of paper by **Nov. 14, 1990,** to ICCI 91, School of Computer Science, Carleton Univ., Ottawa, Canada K1S 5B6, fax (613) 788-4334, e-mail icci@scs.carleton.ca.

SCM 3, Third Int'l Workshop on Software Configuration Management: June 12-14, 1991, Trondheim, Norway. Cosponsors: ACM et al. Submit four copies of position paper and full paper by **Nov. 15, 1990,** to Peter Feiler, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3890, phone (412) 268-7790, e-mail phf@sei.cmu.edu.

CAR 91, Fifth Int'l Symp. on Computer-Assisted Radiology: July 3-6, 1991, Berlin. Sponsor: Technical Univ. Berlin. Submit five copies of abstract by **Nov. 15, 1990,** to Heinz U. Lemke, Inst. for Technical Computer Science, Sekr CG-FR3-3, Franklinstrasse 28-29, D-1000, Berlin 10, Germany, phone 49 (30) 314-73100; or Michael H. Rhodes, Toshiba America MRI, 280 Utah Ave., South San Francisco, CA 94080, phone (415) 875-2909.

IEEE Trans. on Systems, Man, and Cybernetics plans a special issue on distributed artificial intelligence. Submit five copies of manuscript by **Nov. 15, 1990,** to Edmund H. Durfee, EECS Dept., Univ. of Michigan, 1101 Beal Ave., Ann Arbor, MI 48109-2110, phone (313) 936-1563, e-mail durfee @caen.engin.umich.edu.

Third Symp. on Integrated Ferroelectrics: Apr. 3-5, 1991, Colorado Springs, Colo. Submit abstract by **Nov. 15, 1990,** to Conf. Secretary, Microelectronics Research Lab, Univ. of Colorado at Colorado Springs, PO Box 7150, Colorado Springs, CO 80933-7150, phone (719) 593-3488, fax (719) 594-4257.

Parle 91, Conf. on Parallel Architectures and Languages Europe: June 10-13, 1991, Eindhoven, The Netherlands. Cosponsors: Commission of European Communities et al. Submit paper by **Nov. 15, 1990,** to F. Stoots, Philips Research Labs, PO Box 80.000, 5600 JA Eindhoven, The Netherlands, fax 31 (40) 744-758, e-mail stoots@dooma.prl.philips.nl.

KMET 91, Int'l Conf. on Knowledge Modeling and Expertise Transfer: Apr. 22-24, 1991, Sophia Antipolis, France. Cosponsors: Assoc. Francaise pour la Cybernetique Economique et Technique et al. Submit four copies of paper by **Nov. 15, 1990,** to KMET 91, Univ. de Nice, Sophia Antipolis, CNRS, I3S-LISAN, Daniele Herin-Aime, bat. 4, rue A. Einstein, 06560, Valbonne, France, phone (33) 92-94-26-27, fax (33) 92-94-28-98, e-mail dh@cerisi.cerisi.fr.

First Int'l Conf. on Artificial Intelligence in Design: June 25-27, 1991, Edinburgh, Scotland. Submit full paper by **Nov. 16, 1990,** to John Gero, Architectural and Design Science Dept., Univ. of Sydney, NSW 2006, Australia, phone 61 (2) 692-2328, fax 61 (2) 692-3031, e-mail john@archsci.arch.su.oz or john%archsci.su.oz@uunet.uu.net.

FTCS 21, 21st Int'l Symp. on Fault-Tolerant Computing: June 25-27, 1991, Montreal. Sponsor: IEEE Computer Soc. Tech. Committee on Fault-Tolerant Computing. Submit six copies of paper by **Nov. 19, 1990,** to Eduard Cerny, Univ. de Montreal, Departement d'I.R.O., PO Box 6128, Station A, Montreal, Que., Canada H3A 3J7, fax (514) 343-2155, e-mail cerny@iro.umontreal.ca.

Symp. on Experiences with Distributed and Multiprocessor Systems: Mar. 21-22, 1991, Atlanta. Sponsor: Usenix Assoc. Submit 10 copies of full paper by **Nov. 19, 1990,** to Gene Spafford, Software Eng. Research Center, Computer Sciences Dept., Purdue Univ., West Lafayette, IN 47907-2004, phone (317) 494-7825, e-mail spaf@cs.purdue.edu.

Arith 10, 10th Symp. on Computer Arithmetic: June 26-28, 1991, Grenoble, France. Cosponsors: ACM et al. Submit four copies of complete paper by **Nov. 19, 1990,** to David W. Matula, Computer Science and Eng. Dept., Southern Methodist Univ., Dallas TX 75275, phone (214) 692-3080, fax (214) 692-4138, e-mail matula@csvax.seas.smu.edu (for the Americas and the Pacific); or Peter Kornerup, Math. and Computer Science Dept., Odense Univ., DK-5230 Odense, Denmark, phone (45) 66-15-86-00, fax (45) 65-93-26-91, e-mail kornerup@imada (for Europe, Africa, and Asia).

44th Conf. of the Soc. for Imaging Science and Technology: May 12-17, 1991, St. Paul, Minn. Submit abstract and author's application by **Nov. 19, 1990,** to Stanley Busman, Graphic Research Lab, 3M, 3M Center 201-3N-04, St. Paul, MN 55144, phone (612) 733-2664, fax (612) 733-0648.

1991 ACM Symp. on Personal and Small Computers: June 12-14, 1991, Toronto. Cosponsors: Nat'l Research Council of Canada et al. Submit five copies of paper by **Nov. 19, 1990,** to Michael Bauer, Computer Science Dept., Middlesex College, Univ. of Western Ontario, London, Ont., Canada N6A 5B7, e-mail bauer@csd.uwo.ca, bauer@uwovax.bitnet.

ISCA 18, 18th Int'l Symp. on Computer Architecture: May 27-30, 1991, Toronto. Cosponsor: ACM. Submit five copies of manuscript by **Nov. 21, 1990,** to John Hayes, Electrical Eng. and Computer Science Dept., Univ. of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109, phone (313) 763-0386.

Int'l Conf. on Symbolic-Numeric Data Analysis and Learning: Sept. 17-20, 1991, Paris. Sponsor: Institut National de Recherche en Informatique et en Automatique. Submit paper by **Nov. 30, 1990,** to Conf. Secretariat, INRIA, Service des Relations Exterieures, Domaine dc Voluceau — Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France; phone 33 (1)-39-63-55-00, fax 33 (1) 39-63-56-38.

SID 91, 1991 Int'l Symp., Seminar, and Exhibition: May 6-10, 1991, Anaheim, Calif. Sponsor: Soc. for Information Display. Submit abstract and technical summary by **Nov. 30, 1990,** to Jay Morreale, Palisades Inst. for Research Services, 201 Varick St., Suite 1140, New York, NY 10014, phone (212) 620-3371, fax (212) 620-3379.

ECOOP 91, Fifth European Conf. on Object-Oriented Programming: July 15-19, 1991, Geneva, Switzerland. Submit five copies of paper by **Nov. 30, 1990,** to Pierre America, Philips Research Labs, PO Box 80.000, 5600 JA Eindhoven, The Netherlands.

Symp. on Solid Modeling Foundations and CAD/CAM Applications: June 5-7, 1991, Austin, Texas. Sponsor: ACM SIGGraph. Submit five copies of full paper by **Nov. 30, 1990,** to Jaroslaw Rossignac, J2-C03, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, phone (914) 784-7630, fax (914) 784-7455, e-mail jarek@ibm.com.

Int'l J. Computer-Aided VLSI Design plans a special issue on hardware accelerators for computer-aided design. Publisher: Ablex. Submit five copies of full paper by **Dec. 1, 1990,** to Ausif Mahmood, Electrical Eng. Dept., Washington State Univ. at Tri-Cities, 100 Sprout Rd., Richland, WA 99352, phone (509) 375-9234.

1991 ACM Int'l Conf. on Supercomputing: June 17-21, 1991, Cologne, Germany. Cosponsors: Gesellschaft für Informatik et al. Submit five copies of full manuscript by **Dec. 1, 1990,** to Elias Houstis, Computer Science Dept., Purdue Univ., West Lafayette, IN 47907 (for the Americas); Ulrich Trottenberg, GMD/F1, Schloss Birlinghoven, D-5205 Sankt Augustin 1, Germany (for Europe and Africa); or Toshitugu Yuba, ETL 1-1-4 Umesono, Tukuba, Ibaraki, 305 Japan (for Japan and the Far East).

CBMS 91, Fourth IEEE Symp. on Computer-Based Medical Systems: May 12-14, 1991, Baltimore. Cosponsors: IEEE Computer Society, IEEE Engineering in Medicine and Biology Society, and IEEE Baltimore Section. Submit four copies of summary by **Dec. 5, 1990,** to Isaac Bankman, Johns Hopkins Univ., Applied Physics Lab., Bldg. 2-257, Johns Hopkins Rd., Laurel, MD 20723-6099, phone (301) 953-5000, ext. 4253, fax (301) 953-1093.

Eurographics 91, Conf. of the European Assoc. for Computer Graphics: Sept. 2-6, 1991, Vienna. Submit paper by **Dec. 10, 1990,** to Eurographics 91, c/o Interconvention, Austria Center A-1450, Vienna, Austria, phone 43 (1) 2369-2640, fax 43 (1) 2369-648.

Fifth Israel Conf. on Computer Systems and Software Eng.: May 28-29, 1991, Herzlia, Israel. Sponsors: IEEE Computer Soc. Israeli Chapter et al. Submit four copies of extended abstract by **Dec. 14, 1990,** to M. Winokur, c/o ORTRA, PO Box 50432, Tel Aviv 61500, Israel, phone 972 (3) 664-825, fax 972 (3) 660-952.

DCC 91, Data Compression Conf.: Apr. 8-10, 1991, Snowbird, Utah. Sponsor: IEEE Computer Society TCC, NASA/CESDIS. Submit three copies of extended abstract by **Dec. 15, 1990,** to Martin Cohn, Computer Science Dept., Brandeis Univ., Waltham, MA 02254.

Compsac 91, 15th Int'l Software and Applications Conf.: Sept. 11-13, 1991, Tokyo. Cosponsor: Information Processing Society of Japan. Submit six copies of paper by **Jan. 12, 1991,** to Lionel M. Ni, Michigan State Univ., Computer Science Dept., A714 Wells Hall, East Lansing, MI 48824-1027, phone (517) 353-4386, fax (517) 336-1061, Internet ni@cps.msu.edu (for the Americas, Europe, and Africa); or Motoei Azuma, Waseda Univ., c/o Business Center for Academic Societies of Japan, 3-23-1 Hongo, Bunkyo-ku, Tokyo 113, Japan, phone 81 (3) 817-5831, fax 81 (3) 817-5836.

Sixth Int'l Workshop on Software Specification and Design: Oct. 25-26, 1991, Como, Italy. Submit five copies of regular or position paper by **Jan. 21, 1991,** to Carlo Ghezzi, Dip. di Elettronica Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italia, e-mail relett24@imipoli.bitnet.

IEEE Trans. on Computers plans a special issue on artificial neural networks. Submit six copies of manuscript by **Feb. 1, 1991,** to Benjamin W. Wah, Coordinated Science Lab, MC228, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801-3082, phone (217) 333-3516, fax (217) 244-1764, e-mail wah%aquinas@uxc.cso.uiuc.edu.

October 1990

12th Saudi Nat'l Computer Conf. on Planning for the Informatics Soc., Oct. 21-24, Riyadh, Saudi Arabia. Cosponsors: King Saud Univ., Saudi Computer Soc. Contact Mohammad M. Mandurah, College of Computer and Information Sciences, PO Box 51178, Riyadh, 11543, Saudi Arabia, phone 996 (1) 467-6993.

OOPSLA 90, Fifth Conf. on Object-Oriented Programming Systems, Languages, and Applications, Oct. 21-25, Ottawa, Canada. Sponsor: ACM. Contact Assoc. for Computing Machinery, 11 W. 42nd St., New York, NY 10036, phone (212) 869-7440.

FOCS, 31st Foundations of Computer Science, Oct. 22-24, St. Louis. Contact Christos Papadimitriou, Computer Science Dept., Univ. of California at San Diego, La Jolla, CA 92093, phone (619) 534-2086.

Int'l Conf. on Computer Applications in Developing Countries, Oct. 22-24, Benin City, Nigeria. Sponsor: Large Scale Systems Research Group, Univ. of Benin. Contact E.A. Onibere, Math. and Computer Science Dept., Univ. of Benin, P.M.B. 1154, Benin City, Nigeria.

Ninth Nat'l Conf. on EDP System and Software Quality Assurance, Oct. 22-24, Washington, DC. Sponsor: Data Processing Management Assoc. Contact US Professional Development Inst., EDP System and Software Quality Assurance, 1734 Elton Rd., Suite 221, Silver Spring, MD 20903-1733, phone (301) 445-4400, fax (301) 445-5722.

JCIT 5, Fifth Jerusalem Conf. on Information Technology, Oct. 22-25, Jerusalem. Sponsor: Information Processing Assoc. of Israel. Contact Abraham Peled, IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598.

Second Int'l Machinery Monitoring and Diagnostics Conf., Oct. 22-25, Los Angeles. Sponsors: Union College, Schenectady, N.Y.; Soc. for Experimental Mechanics. Contact Union College, Wells House, 1 Union Ave., Schenectady, NY 12308, phone (518) 370-6673, fax (518) 370-6875.

CC 90, Third Int'l Workshop on Compiler Compilers, Oct. 22-26, Schwerin, East Germany. Sponsors: German Democratic Republic Academy of Sciences Inst. of Informatics and Computing Technique et al. Contact Michael Albinus, CC 90 Organizing Committee, Akademie der Wissenschaften der DDR, Inst. fur Informatik und Rechentechnik, Rudower Chaussee 5, Berlin, GDR — 1199.

CALENDAR

In the accompanying Calendar, the IEEE Computer Society logo identifies the conferences the society is sponsoring or participating in. Other conferences of interest to our readers, as well as their sponsors, are also listed.

For inclusion in Call for Papers or Calendar, submit the following information: event name, date(s), location, and sponsor(s) as well as the phone and fax numbers and the electronic address of the person to contact. In addition, for Calls for Papers listings, include the name of the person to whom papers should be submitted and the deadline for submittals.

Computer should receive the above-mentioned information **at least five weeks before the month of publication** (i.e., for the **December 1990** issue, send information for receipt by **October 22, 1990**) to Chuck Governale, Calendar Dept., *Computer*, PO Box 3014, Los Alamitos, CA 90720-1264.

Third Int'l Symp. on Artificial Intelligence, Oct. 22-26, Monterrey, N.L. Mexico. Sponsors: Inst. Tecnologico y de Estudios Superiores de Monterrey et al. Contact Hugo Terashima, Centro de Inteligencia Artificial, ITESM, Suc. de Correos "J", C.P. 64849 Monterrey, N.L. Mexico, phone 52 (83) 58-2000, fax 52 (83) 58-0771, e-mail isai@tecmtyvm.bitnet.

Lecture Series on High-Integrity Systems, Oct. 23, Gaithersburg, Md. Sponsor: Nat'l Inst. of Standards and Technology. Contact Dolores Wallace, NIST, Bldg., B266, Gaithersburg, MD 20899, phone (301) 975-3340, e-mail wallace@swe.swe.ncsl.nist.gov.

Visualization 90, Oct. 23-26, San Francisco. Contact Bruce Brown, Oracle Corp., 20 Davis Dr., Belmont, CA 94002, phone (415) 598-3628.

10th Int'l Workshop on Distributed Artificial Intelligence, Oct. 23-27, Bandera, Texas. Cosponsors: Am. Assoc. for Artificial Intelligence, MCC. Contact Michael N. Huhns, MCC 3500 W. Balcones Center Dr., Austin, TX 78759, phone (512) 338-3651.

ESORICS 90, European Symp. on Research in Computer Security, Oct. 24-26, Toulouse, France. Sponsor: AFCET. Contact Martin Gilles, 16 Para de Diane, 78350 Jouy eu Josas, Toulouse Cedex, France.

First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Oct. 25-26, Kyoto, Japan, and **Oct. 29-31,** Tokyo. Cosponsors: Kansai Inst. of Information Systems et al. Contact John H. Boose, Advanced Technology Center, Boeing Computer Services 7L-64, PO Box 24346, Seattle, WA 98124, phone (206) 865-3253.

Integrated Systems Conf., Oct. 28-31, San Antonio, Texas. Sponsor: Inst. of Industrial Engineers. Contact IIE, 25 Technology Park/Atlanta, Norcross, GA 30092, phone (404) 449-0460, fax (404) 263-8532.

NACLP 90, 1990 North Am. Conf. on Logic Programming, Oct. 28-Nov. 1, Austin, Texas. Cosponsor: ACM. Contact Carlo Zaniolo, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759, phone (512) 338-3442.

Int'l Conf. on Information Technology, Oct. 29-31, Bournemouth, UK. Sponsor: Institution of Electrical Engineers. Contact Conf. Services, IEE, Savoy Place, London WC2R 0BL, UK, phone 44 (71) 240-1871, fax 44 (71) 240-7735.

Eighth Pacific Northwest Software Quality Conf., Oct. 29-31, Portland, Ore. Sponsor: Pacific Northwest Software Quality Conf. Committee. Contact Terri Moore, Pacific Agenda, PO Box 10142, Portland, OR 97210, phone (503) 223-8633.

ISCIA 5, Fifth Int'l Symp. on Computer and Information Sciences, Oct. 30-Nov. 2, Cappadocia, Nevsehir, Turkey. Sponsors: Istanbul Tech. Univ. et al. Contact A. Emre Harmanci, Istanbul Tech. Univ., Bilgi Islem Merkezi, Ayazaga, 80626 Istanbul, Turkey, phone 090 (1) 176-3254, fax 090 (1) 176-1734, e-mail harmanci@tritu.bitnet.

Compsac 90, 14th Int'l Computer Software and Applications Conf., Oct. 31-Nov. 2, Chicago. Contact Ifay F. Chang, Rm. 1B28, IBM T.J. Watson Research Center, PO Box 714, Yorktown Heights, NY 10598, phone (914) 789-7825, fax (914) 784-6211.

November 1990

14th SCAMC, 1990 Symp. on Computer Applications in Medical Care, Nov. 4-7, Washington, DC. Cosponsors: George Washington Univ. Medical Center et al. Contact SCAMC — Office of CEM, George Washington Univ. Medical Center, 2300 K St. NW, Washington, DC 20037, phone (202) 994-8928.

ICCC 90, 10th Int'l Conf. on Computer Comm., Nov. 4-8, New Delhi, India. Sponsor: Int'l Council on Computer Comm. Contact Saroj Chowla, ICCC 90, CMC Ltd., A-5 Ring Rd., South Extension Part I, New Delhi 110049, India, phone 91 (11) 626-807, fax 91 (11) 684-4652.

Optcon 90, Applications in Optical Science and Eng. Conf., Nov. 4-9, Boston. Cosponsors: IEEE Lasers and Electro-Optics Soc. et al. Contact SPIE, PO Box 10, Bellingham, WA 98227-0010, phone (206) 676-3290, fax (206) 647-1445.

1990 IFIP-IEEE Int'l Workshop on Defect and Fault Tolerance in VLSI Systems, Nov. 5-7, Grenoble, France. Contact Gabriel Saucier, Inst. Nat'l Polytechnique de Grenoble/CSI, 46 avenue Felix-Viallet, 38031 Grenoble Cedex, France, phone (33) 76-57-46-87, fax (33) 76-50-23-21; or Tulin E. Mangir, TRW, 1 Space Park, R2/2036, Redondo Beach, CA 90278, phone (213) 813-3894, fax (213) 813-3709.

24th Asilomar Conf. on Signals, Systems, and Computers, Nov. 5-7, Pacific Grove, Calif. Sponsors: Naval Postgraduate School et al. Contact George M. Dillard, Naval Ocean Systems Center, San Diego, CA 92152-5000, phone (619) 553-2478.

Forte 90, Third Int'l Conf. on Formal Description Techniques, Nov. 5-8, Madrid. Contact Juan Quemada, Ingenieria Telematica Dept., ETSI Telecomunicacion, E-28028, Madrid, Spain, phone 34 (1) 549-5700, fax 34 (1) 243-2077, e-mail jquemada@dit.upm.es.

ICCS 90, Int'l Conf. on Comm. Systems, Nov. 5-9, Singapore. Cosponsors: IEEE Singapore Section et al. Contact ICCS 90, c/o Meeting Planners Pte. Ltd., 100 Beach Rd. #33-01, Shaw Towers, Singapore 0718.

Second SIAM Conf. on Linear Algebra in Signals, Systems, and Control, Nov. 5-9, San Francisco. Contact Soc. for Industrial and Applied Math., 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siamconfs@wharton.upenn.edu.

Second NASA Symp. on VLSI Design, Nov. 6, Moscow, Idaho. Contact Judy Wood, NASA SERC, College of Eng., Univ. of Idaho, Moscow, ID 83843, phone (208) 885-6500, fax (208) 885-6645.

Intelligent Robotic Systems: Design and Applications, Nov. 6-7, Philadelphia. Spon-

sor: SPIE. Contact Mohan M. Trivedi, Univ. of Tennessee, Electrical and Computer Eng., Ferris Hall, Knoxville, TN 37996-2100, phone (615) 974-5450.

ACM Conf. on Critical Issues, Nov. 6-7, Arlington, Va. Contact Jim Adams, ACM, 11 W. 42nd St., New York, NY 10036, phone (212) 869-7440, fax (212) 869-1228.

Gomac 90, 1990 Govt. Microcircuit Applications Conf., Nov. 6-8, Las Vegas. Cosponsors: US Defense Dept. et al. Contact Randolph A. Reitmeyer, USALABCOM, Electronics Technology and Devices Lab, ATTN: SLCET-I, Ft. Monmouth, NJ 07703-5000, phone (201) 544-3465.

TAI 90, Second Computer Soc. Int'l Conf. on Tools for Artificial Intelligence, Nov. 6-9, Washington, DC. Cosponsors: Rutgers Univ. et al. Contact Nikolas G. Bourbakis, IBM, 5600 Cottle Rd., San Jose, CA 95193, phone (408) 270-3455.

IEEE Workshop on the Management of Replicated Data, Nov. 7-9, Houston. Sponsor: IEEE Computer Soc. Tech. Committee on Operating Systems. Contact Luis-Felipe Cabrera, IBM Almaden Research Center, 650 Harry Rd., MC K55/803, San Jose, CA 95120-6099, phone (408) 927-1838.

1990 IEEE Workshop on VLSI Signal Processing, Nov. 7-9, San Diego, Calif. Contact Patti Fenstermacher, AT&T Bell Labs, 1243 S. Cedar Crest Blvd., Allentown, PA 18103, e-mail psf@aloft.att.com; or Howard S. Moscovitz, AT&T Bell Labs, 1243 S. Cedar Crest Blvd., Allentown, PA 18103, e-mail mosc@aloft.att.com.

Int'l Workshop on Network and Operating System Support for Digital Audio and Video, Nov. 8-9, Berkeley, Calif. Sponsor: Int'l Computer Science Inst. Contact Ramesh Govindan, ICSI, 1947 Center St., Suite 600, Berkeley, CA 94704-1105, phone (415) 642-4274, ext. 136, e-mail av-workshop@berkeley.edu.

Computational Science in Industry and the Comprehensive Univ., Nov. 8-10, Pomona, Calif. Sponsor: Calif. State Polytechnic Univ. at Pomona. Contact Bruce P. Hillam, Computer Science Dept., Calif. State Polytechnic Univ., 3801 W. Temple Ave., Pomona, CA 91768, phone (714) 869-3440.

Fourth Southeastern Small-College Computing Conf., Nov. 9-10, Hickory, N.C. Sponsor: Consortium for Computing in Small Colleges. Contact Susan Dean, Samford Univ., 800 Lakeshore Dr., Birmingham, AL 35229, Bitnet stdean@samford.bitnet.

ICCAD 90, IEEE Int'l Conf. on Computer-Aided Design, Nov. 12-15, Santa Clara, Calif. Cosponsor: IEEE Circuits and Systems Soc. Contact Pat Pistilli, MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4562 or 4333.

Autofact 90, Robots I4, and Vision 90, Nov. 12-15, Detroit. Cosponsors: Soc. of Manufacturing Engineers and SME Machine Vision Assoc. Contact SME Conf. Dept., PO Box 930, Dearborn, MI, phone (313) 271-1500, ext. 369.

Int'l Conf. on Applications of Software Measurement, Nov. 12-15, San Diego, Calif. Sponsor: Am. Soc. for Quality Control. Contact Software Quality Eng., 3000-2 Hartley Rd., Jacksonville, FL 32257, phone (904) 268-8639.

Supercomputing 90, Nov. 12-16, New York City. Cosponsor: ACM. Contact Joanne L. Martin, IBM T.J. Watson Research Center, PO Box 218, Route 134, Yorktown Heights, NY 10598, phone (914) 945-3285, e-mail jlmart@ibm.com; or Supercomputing 90, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

Second Int'l Symp. on Electronic Art, Nov. 12-17, Groningen, Holland. Sponsor: Groningen Polytechnic. Contact SISEA, Westerhavenstraat 13, 9718 AJ Groningen, Holland, phone 31 (50) 138160, fax 31 (50) 138242.

Seventh Governor's Symp. on High Technology, Nov. 13-15, Kauai, Hawaii. Sponsor: State of Hawaii. Contact William M. Ball, State of Hawaii, 300 Kahelu St., Suite 35, Mililani, HI 96789, phone (808) 625-5293.

PRICAI 90, Pacific Rim Int'l Conf. on Artificial Intelligence 90, Nov. 14-16, Nagoya-shi, Aichi, Japan. Sponsor: Japanese Soc. for Artificial Intelligence et al. Contact Teruo Fukumura, Inter Group Corp., Akasaka Yamakatsu Bldg., 8-5-32 Akasaka, Minato-ku, Tokyo 107, Japan, phone (03) 479-5535.

Third North Carolina Symp. on Artificial Intelligence, Nov. 15-16, Research Triangle Park, N.C. Cosponsors: IEEE Computer Soc. of Eastern North Carolina et al. Contact Connie McElroy-Bacon or Belinda Niedwick, Div. for Lifelong Education, North Carolina State Univ., Box 7401, Raleigh, NC 27695-7401, phone (919) 737-2261.

Cognitiva 90, Nov. 20-23, Madrid. Sponsor: AFCET. Contact Cognitiva 90, c/o Assoc. Francaise pour la Cybernetique Economique et Technique, 156 Bd. Pereire, 75017 Paris, France, phone 33 (1) 4766-2419, fax 33 (1) 4267-9312.

IFIP Workshop on Electronic Design Automation Frameworks, Nov. 26-28, Charlottesville, Va. Sponsor: Int'l Federation for Information Processing. Contact Ron Waxman, Univ. of Virginia, Thornton Hall, Charlottesville, VA 22903, phone (804) 924-6086.

IEEE 1990 Conf. on Software Maintenance, Nov. 26-29, San Diego, Calif. Contact Thomas M. Pigoski, USN, NSGD Pensacola, Corry Station, Pensacola, FL 32511, phone (904) 452-6399.

Micro 23, 23rd Symp. and Workshop on Microprogramming and Microarchitecture, Nov. 27-29, Orlando, Fla. Cosponsor: ACM. Contact Chris Papachristou, Case Western Reserve Univ., Computer Eng. and Science Dept., Cleveland, OH 44106, phone (216) 368-5277, e-mail cap@alpha.ces.cwru.edu.

December 1990

First Int'l Symp. on Uncertainty and Analysis: Fuzzy Reasoning, Probabilistic Methods, and Risk Management, Dec. 3-5, College Park, Md. Sponsors: Univ. of Maryland et al. Contact Bilal M. Ayyub, Civil Eng. Dept., Univ. of Maryland, College Park, MD 20742, phone (301) 405-1956.

ICCV 90, Third Int'l Conf. on Computer Vision, Dec. 4-7, Osaka, Japan. Contact ICCV 90, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

11th Real-Time Systems Symp., Dec. 5-7, Orlando, Fla. Sponsor: IEEE Computer Soc. Tech. Committee on Real-Time Computing. Contact Doug Locke, IBM — MS 409, Systems Integration Div., 6600 Rockledge Dr., Bethesda, MD 20817, phone (301) 493-1496, e-mail cdl@cs.cmu.edu.

CASE 90, Fourth Int'l Workshop on Computer-Aided Software Eng., Dec. 5-8, Irvine, Calif. Contact Elliott J. Chikofsky, Radius Systems, 75 Lexington St., Burlington, MA 01803, phone (617) 494-8200.

WSC 90, 1990 Winter Simulation Conf., Dec. 9-12, New Orleans. Contact Randall P. Sadowski, Systems Modeling Corp., 504 Beaver St., Sewickley, PA 15143, phone (412) 741-3727.

Second IEEE Symp. on Parallel and Distributed Processing, Dec. 9-13, Dallas. Cosponsor: Dallas Chapter of the IEEE Computer Soc. Contact Behrooz Shirazi, Computer Science Dept., Southern Methodist Univ., 6425 Airline Rd., Dallas, TX 75275-0122, phone (214) 692-2874, e-mail shirazi%smu.uucp@uunet.uu.net.

San Diego Workshop on Volume Visualization, Dec. 10-12, La Jolla, Calif. Cosponsor: ACM. Contact T. Todd Elvins, SDSC, Box 85608, San Diego, CA 92038, phone (619) 534-5128.

1990 IEEE Workshop on Languages and Architectures for Automation, Dec. 19-21, Honolulu. Sponsors: Pacific Int'l Center for High Technology Research et al. Contact D.Y.Y. Yun, Univ. of Hawaii, 711 Kapiolani Blvd., Suite 200, Honolulu, HI 96813-5249, phone (808) 539-1532, fax (808) 941-1399; or Shi-Kuo Chang, 322 Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260, phone (412) 624-8493, fax (412) 624-8465, e-mail chang@vax.cs.pitt.edu.

January 1991

Fourth CSI/IEEE Int'l Symp. on VLSI Design, Jan. 5-8, New Delhi, India. Sponsors: Computer Soc. of India et al. Contact Yashwant K. Malaiya, Computer Science Dept., Colorado State Univ., Fort Collins, CO 80523, phone (303) 491-7031, fax (303) 491-2293, e-mail malaiya@ravi.cs.colostate.edu; or D. Roy Chowdhury, Gateway Design Automation, SDF#A-1, Noida Export Processing Zone, PO NEPZ, Noida 201305, India, phone 91 (05736) 62342, fax 91 (05736) 62343.

Int'l Workshop on Formal Methods in VLSI Design, Jan. 9-11, Puerto Rico. Cosponsors: ACM, IFIP. Contact P.A. Subrahmanyam, Rm. 4E-530, AT&T Bell Labs, Holmdel, NJ 07733, phone (201) 949-5812, fax (201) 949-3697, e-mail subra@vax135 .att.com.

Int'l Conf. on Multimedia Information Systems, Jan. 16-18, Singapore. Contact Desai Narasimhalu or Juzar Motiwalla, Inst. of Systems Science, Nat'l Univ. of Singapore, Heng Mui Keng Terr., Kent Ridge, Singapore 0511, phone (65) 772-2075, fax (65) 772-2002, Bitnet issad@nusvm.

PADS, Workshop on Parallel and Distributed Simulation, Jan. 21-23, Anaheim, Calif. Cosponsors: ACM, SCS. Contact David M. Nicol, Computer Science Dept., College of William and Mary, Williamsburg, VA 23185, phone (804) 221-3458, e-mail nicol@cs.wm.edu.

IEEE Int'l Conf. on Wafer Scale Integration, Jan. 29-31, San Francisco. Cosponsors: IEEE Components, Hybrids, and Manufacturing Technology Soc. Contact Terry Chappell, 730 Encino Dr., Aptos, CA 95003, phone (408) 662-1936; or R. Mike Lea, Brunel Univ., Uxbridge UB8 3PH, UK, phone (44) 895-74000, ext. 2821, fax (44) 895-58728, e-mail mike.lea@brunel.ac.uk.

February 1991

CAIA 91, Seventh IEEE Conf. on Artificial Intelligence Applications, Feb. 24-28, Miami Beach, Fla. Contact IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

EDAC 91, European Design Automation Conf., Feb. 25-28, Amsterdam. Sponsor: Institution of Electrical Engineers. Contact Secretariat, EDAC 91, CEP Consultants, 26-28 Albany St., Edinburgh EH1 3QH, Scotland, phone 44 (31) 557-2478, fax 44 (31) 557-5749.

Compcon Spring 91, Feb. 25-Mar. 1, San Francisco. Contact Compcon Spring 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

March 1991

Fifth Int'l Workshop on High-Level Synthesis, Mar. 3-6, Buhlerhohe, West Germany. Cosponsors: IEEE et al. Contact Raul Camposano, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, phone (914) 945-3871, e-mail raulc@ibm.com.

Fourth Computer Virus and Security Conf., Mar. 14-15, New York City. Sponsor: Data Processing Management Assoc. Financial Industries. Contact Judy S. Brand, PO Box 6313, FDR Station, New York, NY 10150, phone (800) 835-2246.

Symp. on Experiences with Distributed and Multiprocessor Systems, Mar. 21-22, Atlanta. Sponsor: Usenix Assoc. Contact George Leach, AT&T Paradyne, MS LG-129, PO Box 2826, Largo, FL 34649-2826, phone (813) 530-2376, e-mail reggie@pdn.paradyne.com.

April 1991

Dasfaa 91, Second Int'l Symp. on Database Systems for Advanced Applications, Apr. 2-4, Tokyo. Sponsor: Information Processing Soc. of Japan. Contact Yahiko Kambayashi, Computer Science Dept., Kyushu Univ., 6-10-1 Hakozaki, Higashi Fukuoka 812, Japan, phone 81 (92) 641-1101, ext. 5407; or Yoshifumi Masunaga, Univ. of Library and Information Science, 1-2 Kasuga, Tsukuba, Ibaraki 305, Japan, phone 81 (298) 52-0511, ext. 340, fax 81 (298) 52-4326, e-mail masunaga@ ulis.ac.jp.

SAC 91, 1991 Symp. on Applied Computing, Apr. 3-5, Kansas City, Mo. Sponsor: Univ. of Missouri — Kansas City. Contact Richard G. Hetherington, SAC 91, Univ. of Missouri — Kansas City, Computer Science Telecommunications Program, 5100 Rockhill Rd., Kansas City, MO 64110-2499, phone (816) 235-2399.

IEEE Infocom 91, Conf. on Computer Comm., Apr. 7-11, Miami, Fla. Cosponsors: IEEE Computer Soc. and Comm. Soc. Contact N. Shacham, IEEE Infocom 91, SRI Int'l, 333 Ravenswood Ave., Menlo Park, CA 94025, phone (415) 859-5710, e-mail shacham@sri.com.

IMS 91, First IEEE Int'l Workshop on Interoperability in Multidatabase Systems, Apr. 8-9, Kyoto, Japan. Contact Ahmed K. Elmagarmid, Purdue Univ., Computer Sciences Dept., West Lafayette, IN 47907, phone (317) 494-1998; or Yutaka Matsushita, Instrumentation Dept., Keio Univ., Hiyoshi, Yokohama, Japan, phone 81 (44) 63-1141, ext. 3564.

DCC 91, Data Compression Conf., Apr. 8-10, Snowbird, Utah. Sponsor: IEEE Computer Society TCC, NASA/CESDIS. Contact John H. Reif, Computer Science Dept., Duke Univ., Durham, NC 27707, phone (919) 660-6568.

ASPLOS 4, Fourth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems, Apr. 8-11, Santa Clara, Calif. Sponsor: ACM. Contact Bob Rau, Hewlett-Packard Labs, 1501 Page Mill Rd., Bldg. 3U, Palo Alto, CA 94304, fax (415) 857-8558, e-mail rau@hplabs.hp.com.

Seventh Int'l Conf. on Data Eng., Apr. 8-12, Kobe, Japan. Contact Ming T. (Mike) Liu, Computer and Information Science Dept., Ohio State Univ., 2036 Neil Ave., Columbus, OH 43210-1277, phone (614) 292-1837, e-mail liu@cis.ircc.ohio-state.edu; or Data Eng. 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013, fax (202) 728-0884.

ETC 91, 1991 European Test Conf., Apr. 10-12, Munich, West Germany. Sponsor: VDE (Zentralstelle Tagungen und Seminare). Contact Peter Stilke, VDE, Stresemannallee 15, D-6000 Frankfurt 70, West Germany, phone (69) 6308-203, fax (69) 6308-273.

Ninth IEEE VLSI Test Symp., Apr. 16-18, Atlantic City, N.J. Cosponsor: IEEE Philadelphia Section. Contact Mukund Modi, Naval Air Eng. Center, ATE Software Center, Code: 52514, Lakehurst, NJ 08733, phone (201) 323-7002, fax (301) 323-7445.

CHDL 91, 10th Int'l Symp. on Computer Hardware Description Languages and their Applications, Apr. 22-24, Marseille, France. Cosponsors: Int'l Federation for Information Processing et al. Contact Dominique Borrione, Imag/Artemis, BP 53X, 38041 Grenoble Cedex, France, phone (33) 7651-4604, ext. 5240, fax (33) 7651-9637, e-mail borrione@imag.imag.fr.

Second Int'l Conf. on Systems Integration, Apr. 22-25, Morristown, N.J. Cosponsors: New Jersey Inst. of Technology et al. Contact Peter A. Ng, Computer and Information Science Dept., New Jersey Inst. of Technology, University Heights, Newark, NJ 07102, phone (201) 596-3387, e-mail ng_p@vienna.njit.edu.

NCGA 91, 1991 Nat'l Computer Graphics Assoc. Conf., Apr. 22-25, Chicago. Contact NCGA, 2722 Merrilee Dr., Suite 200, Fairfax, VA 22031, phone (703) 698-9600.

CHI 91, 1991 Conf. on Human Factors in Computing Systems, Apr. 27-May 2, New Orleans. Sponsor: ACM. Contact Keith Butler, Boeing, Advanced Technology Center, PO Box 24346, M/S 7L-64, Seattle, WA 98124, phone (206) 865-3389; or June Davis, 13 Annapolis St., Annapolis, MD 21401, phone (301) 269-6801.

Int'l Conf. on Cognitive Sciences, Apr. 29-May 2, Montreal. Cosponsors: AFCET et al. Contact Gilles Gauthier, Math.

SECOND PHYSICAL DESIGN WORKSHOP

Module Generation and Silicon Compilation

Nemacolin Woodlands, Laurel Highlands, Pennsylvania, USA

May 20-22, 1991

Sponsored by ACM's SIGDA

FIRST CALL FOR PARTICIPATION

You are invited to participate in the Second Physical Design Workshop to be held at The Nemacolin Woodlands Conference Center in the Laurel Highlands outside Pittsburgh, Pennsylvania on May 20-22, 1991. Presentations will focus in IC physical design with emphasis on recent developments in module generation and silicon compilation including:

- layout from netlists
- layout to/from schematics
- layout from behavioral HDL's
- performance driven layout
- experiences with silicon compilation
- layout benchmark results

The workshop will be limited in the number of attendees.

To facilitate comparison of results a set of standard benchmark circuits will be distributed by December 1990. Please contact the Benchmark Chair to obtain more information about the benchmarks for the workshop.

Participants wishing to deliver a presentation should submit 14 copies of a manuscript whose length does not exceed 3000 words, apart from figures, indicating the status of the work and significant results. Proposals will be reviewed by the Technical Program Committee. Proceedings will be published and distributed at the workshop. **The submission deadline is January 7, 1991.**

WORKSHOP DEADLINES

Jan. 7, 1991	Last day to receive proposals
Feb. 11, 1991	Notification of acceptance
March 11, 1991	Camera-ready copy due
May 1, 1991	Last day to receive proposals for presentation of benchmark results

For further information contact:

Workshop Chair
Antun Domic
HLO2-3/J3
DEC
77 Reed Rd.
Hudson, MA 01749
domic@cadsys.dec.com

Program Chair
Mary Jane Irwin
333 Whitmore Lab
Dept. of Cmp. Science
Penn State Univ.
Univ. Park, PA 16802
mji@cs.psu.edu

Benchmark Chair
Dwight Hill

AT&T Bell Labs
600 Mountain Ave.
Murray Hill, NJ 07974
dwight@allegra.att.com

Arrangements Chair
Steven Levitan
348 Benedum Hall
Dept. of Elec. Engr.
Univ. of Pittsburgh
Pittsburgh, PA 15261
levitan@ee.pitt.edu

and Computer Science Dept., Univ. of Quebec, PO Box 8888, Station A, Montreal, Que., Canada H3C 3P8, phone (514) 987-8212, fax (514) 987-8477.

May 1991

CBMS 91, Fourth IEEE Symp. on Computer-Based Medical Systems, May 12-14, Baltimore. Cosponsors: IEEE Computer Society, IEEE Engineering in Medicine and Biology Society, and IEEE Baltimore Section. Contact Jeffery C. Lesho, Johns Hopkins Univ., Applied Physics Lab., Bldg. 2-257, Johns Hopkins Rd., Laurel, MD 20723-6099, phone (301) 953-5000, ext. 8057.

ICSE 13, 13th Int'l Conf. on Software Eng., May 13-17, Austin, Texas. Cosponsor: ACM. Contact ICSE 13, Bryan Fugate, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759-6509, phone (512) 338-3330; MCC, PO Box 200015, Austin, TX 78720-0015; or ICSE 13, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

CompEuro 91, IEEE Int'l Conf. on Advanced Computer Technology, Reliable Systems, and Applications, May 13-17, Bologna, Italy. Cosponsors: IEEE Region 8 et al. Contact Vito Monaco, Dip. Eletronica Informatica E Sistemistica, Univ. Di Bologna, Viale Risorgimento, I-60136, Bologna, Italy.

CCW 91, Third IEEE Conf. on Computer Workstations, May 15-17, Falmouth, Mass. Sponsor: IEEE Computer Soc. Tech. Committee on Operating Systems. Contact Luis-Felipe Cabrera, IBM Almaden Research Center, MC K55/801, 650 Harry Rd., San Jose, CA 95120-6099, phone (408) 927-1838, e-mail cabrera@ibm.com; or Kenneth Kane, Boston Development Center, Sun Microsystems, 2 Federal St., Billerica, MA 01802, phone (508) 671-0367, e-mail kkane@east.sun.com.

Int'l Symp. on Software Reliability Eng., May 17-18, Austin, Texas. Cosponsors: IEEE Computer Soc. Tech. Committee on Software Eng. and the Nat'l Aeronautics and Space Administration. Contact John C. Munson, Computer Science Div., Univ. of West Florida, Pensacola, FL 32514, phone (904) 474-2989, e-mail jmunson@dcsnet.uwf.edu.

ICDCS 91, 11th Int'l Conf. on Distributed Computing Systems, May 20-24, Arlington, Texas. Contact Bill D. Carroll, Computer Science Eng. Dept., Univ. of Texas at Arlington, Box 19015, Arlington, TX 76019-0015, phone (817) 273-3785, e-mail carroll@evax.ari.utexas.edu.

SESAW, Fourth Software Eng. Standard Application Workshop, May 21-23, San Diego, Calif. Contact Vera V. Edelstein, Nynex, 500 Westchester Ave., White Plains, NY 10604, phone (914) 683-2888.

ISCA 18, 18th Int'l Symp. on Computer Architecture, May 26-30, Toronto. Cosponsor: ACM. Contact K.C. Smith, Univ. of Toronto, Electrical Eng. Dept., Toronto, Ont. M5S 1A4, Canada, phone (416) 978-5033.

Fifth Israel Conf. on Computer Systems and Software Eng., May 28-29, Herzlia, Israel. Sponsors: IEEE Computer Soc. Israeli Chapter et al. Contact M. Winokur, c/o ORTRA, PO Box 50432, Tel Aviv 61500, Israel, phone 972 (3) 664-825, fax 972 (3) 660-952

June 1991

Fourth Int'l Conf. on Industrial and Eng. Applications of Artificial Intelligence and Expert Systems, June 2-5, Kauai, Hawaii. Sponsors: ACM et al. Contact Moonis Ali, Univ. of Tennessee Space Inst., MS15, B.H. Goethert Pkwy., Tullahoma, TN 37388-8897, phone (615) 455-0631, ext. 236, fax (615) 454-2354, e-mail alif@utsiv1.bitnet.

CVPR 91, IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition, June 3-7, Lahaina, Maui, Hawaii. Contact Shahriar Negahdaripour, Electrical Eng. Dept., Univ. of Hawaii at Manoa, 2540 Dole St., Honolulu, HI 96822, e-mail shahriar@wiliki.eng.hawaii.edu.

SCM 3, Third Int'l Software Configuration Management Workshop, June 12-14, Trondheim, Norway. Cosponsors: ACM, et al. Contact Reidar Conradi, Computer Systems and Telematics Div., Norwegian Inst. of Technology, N-7034 Trondheim, Norway, phone 47 (7) 593-444; or Peter Feiler, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3890, phone (412) 268-7790, e-mail phf@sei.cmu.edu.

DAC 91, 28th ACM/IEEE Design Automation Conf., June 16-21, San Francisco. Contact MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4333.

FTCS 21, 21st Int'l Symp. on Fault-Tolerant Computing, June 25-27, Montreal. Sponsor: IEEE Computer Soc. Tech. Committee on Fault-Tolerant Computing. Contact Vinod K. Agarwal, McGill Univ., Electrical Eng. Dept., 3480 University St., Montreal, Que., Canada H3A 2A7, phone (514) 398-7136, fax (514) 398-4470, e-mail agarwal@spock.ee.mcgill.ca.

Arith 10, 10th Symp. on Computer Arithmetic, June 26-28, Grenoble, France. Cosponsors: ACM et al. Contact Jean-Michel Muller, Lab. LIP-IMAC, Ens. Lyon, 69364 Lyon Cedex 07, France, phone 33 (72) 72-8229.

July 1991

CAR 91, Fifth Int'l Symp. on Computer-Assisted Radiology, July 3-6, Berlin. Sponsor: Tech. Univ. Berlin. Contact Heinz U. Lemke, Inst. for Tech. Computer Science, Sekr CG-FR3-3, Franklinstrasse 28-29, D-1000, Berlin 10, Germany, phone 49 (30) 314-73100; or Michael H. Rhodes, Toshiba America MRI, 280 Utah Ave., South San Francisco, CA 94080, phone (415) 875-2909.

August 1991

SSD 91, Second Symp. on Large Spatial Databases, Aug. 28-30, Zurich, Switzerland. Contact H.J. Schek, Inst. für Information Systeme, Eth Zentrum, 8092 Zurich, Switzerland, phone 41 (1) 254-7240.

September 1991

17th Int'l Conf. on Very Large Data Bases, Sept. 3-6, Barcelona, Spain. Sponsors: IEEE Computer Soc. Tech. Committee on Data Eng. et al. Contact Guy Lohman, IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120, e-mail lohman@ibm.com.

Compsac 91, 15th Int'l Computer Software and Applications Conf., Sept. 11-13, Tokyo. Cosponsor: Information Processing Soc. of Japan. Contact Stephen S. Yau, Univ. of Florida, CIS Dept., Rm. 301, Gainesville, FL 32611, phone (904) 335-8006.

October 1991

IEEE Workshop on Visual Motion, Oct. 6-9, Princeton, N.J. Contact Thomas S. Huang, Coordinated Science Lab, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 333-6912.

CSEE 91, Fifth SEI Conf. on Software Eng. Education, Oct. 7-8, Pittsburgh. Contact James E. Tomayko, Software Engineering Inst., Carnegie Mellon Univ., 4500 Fifth Ave., Pittsburgh, PA 15213-3890, phone (412) 268-6806, fax (412) 268-5758.

Workshop on Experimental Distributed Systems, Oct. 12, Huntsville, Ala. Contact Raif M. Yanney, TRW, 1 Space Park, DH2/2328, Redondo Beach, CA 90278, phone (213) 764-6033.

11th Symp. on Mass Storage Systems, Oct. 13-17, Monterey, Calif. Contact Bernard T. O'Lear, NCAR, PO Box 3000, Boulder, CO 80307, phone (303) 497-1268.

RIDT 91, Second Int'l Workshop on Raster Imaging and Digital Typography, Oct. 14-15, Boston. Contact Robert A. Morris, Math and Computer Science Dept., Univ. of Massachusetts at Boston, Harbor Campus, Boston, MA 02125-3393, phone (617) 287-6466.

CALL FOR PAPERS

IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS & PROCESSORS

Sponsored by: IEEE Computer Society and IEEE Circuits and Systems Society
In Cooperation with: IEEE Electronic Devices Society

ICCD '91

Hyatt Regency Cambridge, Cambridge, Massachusetts
October 14-16, 1991

The International Conference on Computer Design: VLSI in Computers and Processors covers all aspects of the design and implementation of VLSI computer and processor systems. The multi-disciplinary nature of the conference is intended to emphasize the interactions among architecture, computer-aided design, design & test, and VLSI technology. Original papers are especially solicited in the following areas as they relate to computer and processor design:

Architecture and Algorithms: Computer Architecture: Concurrent Computers • Digital • Signal and Image Processors • Data Base Machines • Graphics Processors • Architectural Support for Operating Systems and Languages • Computer Design: Cache and Memory System Design • Processor Design • Computer Arithmetic • Computer Networks • Algorithms: Design and Analysis of Algorithms • Parallel Algorithms • Numerical Methods

Computer-Aided Design: High-Level Synthesis • Silicon Compilation • Automatic Layout: Placement and Routing • Layout Verification • Timing Analysis • Logic and Circuit Simulation • Multiprocessor and Parallel Processor • Implementation of CAD Algorithms • Dedicated CAD Hardware • Integrated CAD Systems

Design & Test: Mixed Analog/Digital Design • Design for Testability • Design for Self Test • Testing and Testability Analysis • Fault Modeling • Reliable Computing • Design for Reliability

VLSI & Technology: State of the Art System Integration • Packaging • Wafer-Scale Integration • Environmental Factors • Optical Interconnects • VLSI: CMOS • Bipolar • GaAs • Low Temperature • Mixed Technologies • Storage: Optical • Magnetic • Semiconductor

Papers are also solicited which describe innovative features of new products and the overall integration of Technology, Architecture, CAD, and Design and Test into the Computer Design Process.

Instructions to Authors

Prospective authors are invited to submit a 3-4 page, single spaced summary plus additional pages for key figures, diagrams, and references. To be considered for review, the following information must be included: (a) a clear description of the contribution and why it is important, (b) original research submissions must state what is novel about the contribution, (c) review and tutorial submissions must state the contribution to the multi-disciplinary mission of the conference. Authors should indicate the appropriate technical area for their submission: Architecture and Algorithms, Computer-Aided Design, Design and Test, or VLSI and Technology.

Submit six (6) copies of summaries, along with the authors' names, addresses, e-mail addresses and office and home telephone numbers, no later than **February 1, 1991** to the Technical Program Chair:

Dwight Hill
AT&T Bell Laboratories
3D-446
Murray Hill, NJ 07974
Telephone: 201-582-7766
E-mail: dwight@research.att.com

- Proposals for specially organized sessions are solicited. Please forward your proposals to the Technical Program Chair for review no later than **January 2, 1991.**
- Summaries to Technical Program Chair: **February 1, 1991.**
- Notification of acceptance: **March 31, 1991.**
- Final camera-ready paper due **June 15, 1991.**
- Awards will be presented to the best conference papers.

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
IEEE

❑ **PLEASE SEND ME MORE INFORMATION ABOUT ICCD '91.**

Name _____
Company _____
Address _____
City/State/Zip _____
Country _____

Send to: ICCD '91
1730 Massachusetts Avenue, N.W.
Washington, DC 20036-1903

CAREER OPPORTUNITIES

RATES: $12.00 per line, (ten lines minimum). Average five typeset words per line, eight lines per column inch. Add $10 for box number. Send copy at least one month prior to publication date to: **Marian B. Tibayan, Classified Advertising, COMPUTER Magazine, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.**

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, *COMPUTER* may reject any advertisement containing any of these phrases or similar ones: "...recent college grads...," "...1-4 years maximum experience...," "...up to 5 years experience," or "...10 years maximum experience." *COMPUTER* reserves the right to append to any advertisement, without specific notice to the advertiser, "Experience ranges are suggested minimum requirements, not maximums." *COMPUTER* assumes that, since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only.

ENGINEER, PRODUCT QUALITY ENGINEERING

Company in Denver, CO specializing in development of CAD/CAM and desktop publishing systems seeks Engineer, Product Quality Engineering to work with marketing, development and/or customers to establish and confirm software requirements. Follow projects through all phases of CAD/CAM software development and related applications, including requirements analysis, design, coding, software and hardware testing, and product delivery. Perform testing activities using C, C-Shell, C++, FORTRAN, and Assembly languages on UNIX, Domain/OS and VMS operating systems. Develop internal software quality program and devise software engineering standards. Requires B.S. in Electrical Engineering and Computer Science or Electrical and Computer Engineering; nine months experience in development of software testing tools and software testing; working knowledge of C, C-Shell, C++, and FORTRAN; working knowledge of UNIX, Domain/OS and VMS operating systems; knowledge of internals of operating systems and hardware architectures, capabilities and limitations; knowledge of software metrics, error tracking, data collection methods and feedback analysis, verification and validation. $29,985/year; 8:00 a.m - 5:00 p.m., M-F. Respond by resume to Colorado Department of Labor & Employment, Division of Employment & Training, 600 Grant St., #900, Denver, CO 80203, ATT: Jim Shimada, and refer to Job Order No. CO3194015.

INDIANA UNIVERSITY-PURDUE UNIVERSITY AT INDIANAPOLIS (IUPUI)

The Department of Computer and Information Science invites applications for a tenure track position at the level of assistant or associate professor in scientific computing/software engineering. The successful candidate is expected to be a member of a new interdisciplinary group in computational science, to be developed jointly by the Department of Computer and Information Science and the Department of Mathematical Sciences.

Applicants must have an earned doctorate and strong teaching credentials A demonstrated record of research is expected at the associate professor level. Assistant professor candidates must possess exceptional research potential. Special consideration will be given to applicants with a strong background in numerical computation with expertise and interest in the development of "intelligent" software for large-scale scientific computing problems. Salary and rank will be determined by experience and qualification.

IUPUI is a rapidly growing comprehensive urban university with over 26,000 students. The university offers competitive salaries and provides excellent fringe benefits. Indianapolis is a large, active metropolis and has available a dynamic industrial base conducive to joint research and consulting with our faculty. The six-hospital Medical Center at IUPUI is yet another source of interdisciplinary research for the Department.

Applications and inquiries should be addressed to Prof. Raymond C. Y. Chin, Chair, Department of Computer and Information Science, IUPUI, 1201 East 38th Street, Indianapolis, IN 46205-2868. Deadline for applications: December 1, 1990. Late applications will be considered until position is filled. IUPUI is an Equal Opportunity/Affirmation Action Employer. Women and minority candidates are encouraged to apply.

PURDUE UNIVERSITY
Computer Engineering Faculty Positions

The School of Electrical Engineering at Purdue University seeks outstanding candidates in all areas of Computer Engineering for research and teaching. Openings are for tenure-track faculty at all levels. Active research areas include artificial intelligence and expert systems; computer communications networks; computer vision; design automation tools; digital signal processor system design; distributed algorithms and databases; fault-tolerant and testable computing; microprocessor design; natural language processing; neural networks; parallel processing (architecture, algorithms, operating systems, compiling, languages, interconnection networks, and performance); robot vision, sensors and control; software engineering; speech processing; and VLSI architecture.

The School has 69 full-time faculty (25 in Computer Engineering) and over $9 million in funded research projects. In addition to the PhD, MSEE and BSEE degrees, the School offers a BSCEE (Bachelor of Science in Computer and Electrical Engineering) which is accredited in both computer engineering and electrical engineering. Computing facilities available to the faculty include the Engineering Computer Network (including 10 VAX 11/780's running UNIX BSD 4.3, 1 Gould PN 9080, 4 Gould NP 1's, a CCI PN 6/32, and 400 Sun workstations), the Computing Center's Cyber-205 and ETA-10 supercomputers, a Symbolics LISP machine, an IBM 3090/180E, extensive graphics facilities, and numerous PC's. Parallel computing facilities include a 128-node Ncube hypercube, a 48×48 processor NCR-GAPP processor array, an 82-processor AT&T Pixel Machine, a 16-processor Transputer array, the 30-processor PASM Parallel Processor prototype that was developed and built at Purdue, and the Computing Center's 4 Sequent Symmetry S81. Custom VLSI chip design facilities include Mentor Graphics software running on Apollo Workstations. Equipment in the Robot Vision Lab and Computer Vision and Image Processing Lab includes a Puma 762 robot, a Cincinnati Milacron T3-726 robot, a K2A Cybermation mobile robot. Image processing systems include Gould/DeAnza, Comtal, Grinnell, Imaging Technologies.

Applicants must possess a doctorate degree. Send a resume, including a statement of teaching and research interests and a list of three (3) references, to: Head, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907. Purdue University is an Equal Opportunity/Affirmative Action employer.

UNIVERSITY OF NEVADA, RENO
Chair, Computer Science

Applications are invited for the new position of Chair, Department of Computer Science, University of Nevada, Reno. This individual will not only provide direction in the continuing improvement and development of the academic program, but also strong leadership in the establishment of a significant research program. A Ph.D. in computer science, computer engineering or a related discipline is expected, and the applicant should demonstrate an outstanding record of teaching and research.

Write a letter of application including names of four references and a detailed resume to Prof. Allen Brady, Chair Search Committee, Department of Computer Science, Mackay School of Mines, University of Nevada, Reno, NV 89557 (email: al@tahoe.unr.edu).

The position commences July 1, 1991. Applications will be accepted until December 31, 1990 or subsequently until the position is filled.

Affirmative Action/Equal Opportunity Employer.

UNIVERSITY OF CALIFORNIA AT IRVINE
Faculty Positions in Computer Science

The Department of Information and Computer Science (ICS) is actively recruiting faculty AT ALL LEVELS. We have dynamic research groups in the areas of computer systems design, parallel processing, artificial intelligence, computer networks and distributed processing, software, social and managerial analysis of computing, and theory. We are continuing to build on these areas of strength. We are also interested in developing new strength in computational biology, integrated systems, computer-supported cooperative work, databases, design tools, and model-based reasoning. We will sympathetically review applications from very strong candidates in all areas of computer science.

We are looking for new faculty with strong research records who would thrive in a highly productive but friendly setting, and who would like to join us in exploring the nature of computing, broadly defined. We specially encourage application from exceptionally distinguished candidates for senior positions.

The ICS Department is an independent academic unit reporting to the Executive Vice Chancellor. ICS faculty emphasize core computer science as well as research in emerging areas of the discipline, with effective interdisciplinary ties to colleagues in neurobiology, cognitive science, management, engineering, and the social sciences. The department currently has 30 full-time faculty positions and over 120 Ph.D. students, with major support from the administration to expand and to strengthen the research environment.

Annual research funding from contracts and grants from agencies such as DARPA, NSF, and ONR, currently total over $6.5 million. In 1986 the software group was awarded a Coordinated Experimental Research (CER) grant from the National Science Foundation. This support has fostered the creation of a Research Laboratory for software engineering, in which major studies of the development and evaluation of software technology are undertaken. High quality research has also fostered the creation of a Research Laboratory for computer systems design that deals with methodologies and tools for the design of complex computational systems. A third Research Laboratory, in Artificial Intelligence, is planned.

Department equipment includes approximately 175 workstations, primarily Sun-3's and Sun-4s. Two large multiprocessor Sequents and a Hypercube are available, as well as approximately 300 Macintosh Plus's and II's. All our major workstations and computers are tied together with networks, which are gatewayed to the campus network, and from there, to regional, national, and international networks (Darpa Internet, CSnet, Bitnet, etc.). In addition, department members have access to campus-wide computing resources as well as regional supercomputer access.

UC-Irvine is located in Orange County, three miles from the Pacific Ocean near Newport Beach, and approximately halfway between Los Angeles and San Diego. The campus is situated in the heart of a national center of high-technology enterprise. It is growing rapidly and offers exciting professional and cultural opportunities. Salaries and benefits are competitive. Special housing assistance is available from the university, including newly built, for-sale housing within short walking distance from the Department.

Send resumes and names of four references to:

Chair, Faculty Recruiting Committee
Department of Information and Computer Science
University of California-Irvine
Irvine, CA 92717

The ICS Department has several vacant positions and application screening will begin immediately upon receipt of curriculum vitae. Maximum consideration will be given to applications received by January 31, 1991.

The University of California is an Affirmative Action/Equal Opportunity Employer. The Department of ICS is particularly interested in receiving applications from women and minority candidates.

THE UNIVERSITY OF CINCINNATI
Electrical and Computer Engineering Department

Applications are solicited for several new tenure track faculty positions at all ranks starting September 1, 1991. Applicants in all areas of computer science and engineering are invited to apply, but the following areas are of special interest: microprocessor design, parallel and distributed computing, software engineering, operating systems, programming languages, VLSI systems, computer-aided design, databases, compiler design, architecture, optical computing, and computer vision.

The Department offers MS/Ph.D. program in computing sciences and computer engineering and also offers ABET accredited BS degree in computer engineering. Our Computer Science and Engineering faculty members are actively involved in research in architecture, parallel and distributed systems, operating systems, VLSI design, and computer vision. Departmental resources include a network of SUN Workstations, a VAX 11/785, a VAX 11/750, a 16-processor ES-Kit, and numerous PCs. Every faculty office has a Sun Workstation and laser printer; graduate student offices also have Sun Workstations. The department has recently filled 14 faculty positions and is engaged in an exciting period of further growth. Currently the department has 30 full-time faculty, 140 full-time graduate students, 400 undergraduate students, 2 major research centers, and 19 full-time staff members.

All candidates should have a strong commitment to excellence in research and teaching and an earned Ph.D. in either Computer Science or Computer Engineering. Please send curricula vitae and the names of three references to: Dr. Vik Kapoor, Head, Electrical and Computer Engineering Department, Mail Location 30, University of Cincinnati, Cincinnati, Ohio 45221-0030. E-mail inquiries should be sent to vkapoor@ucecel.ece.uc.edu.

The University of Cincinnati is an Affirmative Action/Equal Opportunity employer and encourages and welcomes applications from women and minorities.

CASE INSTITUTE OF TECHNOLOGY
NORD Professorship in Computer Engineering and Science

The Department of Computer Engineering and Science at Case Institute of Technology is seeking a nationally recognized scholar and researcher to fill the NORD Professorship. This position was recently established by the donation of over one and a half million dollars, which will provide outstanding professional opportunities and a highly competitive salary, together with additional funds for travel, graduate student support and equipment. The qualifications include a Ph.D. in computer science, computer engineering or closely allied fields, and an ability to establish and develop external support for a nationally recognized research program in computer science/computer engineering. We invite applications from senior faculty at both the associate professor and full professor levels.

CWRU is a private university with a total enrollment of 8,400, of which 5,100 are graduate and professional students. The Engineering School of Case Institute of Technology is among the top ten engineering schools in terms of research funding per faculty member and undergraduate student quality. The University campus is the hub of the pleasant area known as University Circle, an incorporation with neighboring cultural centers and museums, about five miles from downtown Cleveland.

The Department of Computer Engineering and Science has 14 faculty positions, and a graduate student body of 110 students, 40 of which are in the Ph.D program. Departmental facilities are based upon a ethernet local area network, connected to INTERNET, which supports a UNIX operating system and about 40 SUN and other workstations. In addition, faculty and students participating in the Center for Automation and Intelligent Systems Research have access to the Center's computing facilities.

Applicants should submit their curriculum vitae and names of at least five references to: Lee J. White, Chairman, **Department of Computer Engineering and Science,** Case Western Reserve University, Cleveland, Ohio 44106; INTERNET: leew@alpha.ces.cwru.edu; applicants may wish to provide at most three reprints of their most important publications.

An equal employment and affirmative action employer.

THE AMERICAN UNIVERSITY IN CAIRO
Faculty Position

One Assistant, Associate, or Full Professor needed to teach, in English, undergraduate courses in Computer Science including theory of computing and design of algorithms. Ph.D. required. Rank, salary based on qualifications and experience. Two-year appointment (renewable) begins September 1991. For expatriates, housing, roundtrip air travel, plus schooling for two children included. Write with curriculum vitae to: Dean George H. Gibson, The American University in Cairo, 866 United Nations Plaza, Suite 517, New York, New York 10017, preferably before December 15.

POLYTECHNIC UNIVERSITY
Head, Department of
Computer Science

The Department of Computer Science of the School of Electrical Engineering and Computer Science invites applications for the position of Head of Computer Science.

Polytechnic University is a technological urban university established in 1854 and located on three campuses in the greater New York area. The university has an enrollment of approximately 4,700 students. Because of its location, there is close interaction with major companies in New York and New Jersey. The main campus is in Downtown Brooklyn adjacent to Brooklyn Heights, one of New York's desirable residential communities. The other two campuses are in Farmingdale, Long Island, and in Hawthorne, Westchester County. The Brooklyn campus is undergoing major development involving Metrotech, a 16-acre academic/research/industrial park. As part of this project, the entire campus will be renovated; in particular, the Computer Science Department will be housed in a new building which is already under construction.

The Department of Computer Science has 20 tenure-track faculty members and offers degree programs leading to the BS, MS and Ph.D. degrees. Active research is being carried out in the areas of software engineering, algorithms, programming languages, compilers, computer architecture, operating systems network analysis and management, and image analysis and understanding. The undergraduate program is accredited by CSAB. In 1989 there were 158 MS degrees and 7 Ph.D. degrees awarded in Computer Science. Computer facilities include Sun workstations, Apollo workstations, a VAX 780, an IBM 4381, and IBM 4341, a Gould 9080, and AT&T 3B2/1000, two AT&T 3B2/500s, two AT&T 3B2/400s, several PC labs, and an instructional laboratory. The campuses are fully networked.

Applicants should have a Ph.D. degree and an outstanding record in Computer Science. Applications and nominations should be sent to the Chairman of the Committee, Professor Theodor Tamir, Department of Electrical Engineering, Polytechnic University, 333 Jay Street, Brooklyn, NY 11201. Polytechnic is an Equal Opportunity Employer, M/F/V/H.

GEORGE MASON UNIVERSITY
Faculty Positions in Computer Science

The Department of Computer Science at George Mason University is seeking applicants for positions at two ranks: assistant professor and full professor. At the assistant professor rank we are interested in recruiting persons in image processing, computer vision, and artificial intelligence to work with growing efforts underway in these areas. We are especially interested in outstanding teachers and researchers in software engineering—artificial intelligence and parallel computing—image processing. At the full professor level we are interested in recruiting an established person with an international reputation in software systems or computer networking-distributed systems to lead a new initiative. The professor rank carries tenure. Visiting faculty positions may be available at all ranks. The nominal date of appointments commences September 1991.

Candidates for these positions must have an earned doctorate in a relevant field, along with professional accomplishments that are appropriate in the rank sought. They should be interested in both teaching and developing a strong sponsored research program.

The relevant research and teaching areas of interest in Computer Science are: artificial intelligence, computer vision and image processing, parallel computation and distributed computation, and software engineering.

There are nearly 100 faculty in SITE, and nearly 25 faculty in Computer Science. There are numerous opportunities for government and industrial interaction in the suburban high technology area, just 15 miles southwest of the Nation's Capital. George Mason University is located in Fairfax County, Virginia, near the outstanding scientific, cultural, and touristic attractions of Washington, D.C. The University has made a major commitment to establish a School of Information Technology and Engineering (SITE) and maintain an outstanding academic and research program in the Department of Computer Science.

The Computer Science Department plans to move into a new building with new advanced computing facilities in 1991. Present office systems include Sun networks, Mac networks, and Vax networks, with research machines including hypercubes, symbolics systems, transputer systems, and NEXT machines.

For full consideration please send a detailed resume together with the names of four references and other supporting material to Professor David Rine, Chair, Recruitment Committee, Department of Computer Science, George Mason University, Fairfax, VA 22030, or call at (703) 323-2713 (e-mail: drine@gmuvax.gmu.edu). Applications are sought by Feb. 1, 1991, and these will receive first priority. AA/EOE.

UNIVERSITY OF FLORIDA
Computer and Information Sciences
Department

Applicants with strong research capability in one of the specialized areas and able to teach a variety of courses in Computer and Information Sciences are invited for tenure-track and visiting positions at all levels. Areas of specialization include software engineering, computer vision, database systems and artificial intelligence. Positions are available starting in Spring 1991 as well as the 1991-92 academic year. Each applicant must have a doctoral degree in Computer Science or a related area or will complete the degree before the start of the faculty appointment.

Applicants should send their resumes and four names and addresses to: Professor Randy Chow, Chairman, Faculty Search and Screening Committee, Computer and Information Sciences Department, 301 CSE, University of Florida, Gainesville, FL 32611; telephone number: (904) 392-1200, e-mail address: chow@cis.ufl.edu.

Closing date: October 31, 1990 or until positions are filled. University of Florida is an Equal Opportunity/Affirmative Action Employer. This faculty search will be in compliance with Florida's "government in the Sunshine Law."

SENIOR ANALYST

Senior Analyst needed to design, implement, and test software packages to be used for production planning and scheduling for large manufacturing companies. Confer with program management to identify customer scheduling needs. Interface with customers in order to design and implement software that will meet their needs. Provide training and technical support for scheduling programs. Analyze, test, and debug scheduling software. Evaluate software for potential scheduling problems. Plan and prepare technical reports and instructional manuals relative to the operation of the program. Requires a Master's degree in Operation Research/Computer Science or its equivalent. Must have at least three credit hours in the following: operations research, computer science, artificial intelligence and scheduling as well as LISP and C programming languages. 40 hour work week. $36,000 per year. Apply at the Texas Employment Commission, Dallas, Texas, or send resume to the Texas Employment Commission, TEC Building, Austin, Texas 78778. Job Order #5424746. Ad paid for by an Equal Employment Opportunity Employer.

CARLETON UNIVERSITY
School of Computer Science
Faculty Position

Applications are invited for a tenure track position at the rank of Assistant or Associate Professor in the School of Computer Science starting July 1, 1991. The School has 17 full-time faculty with particular research strengths in algorithms and complexity, intelligent systems, object-oriented programming, and parallel and distributed computing. Outstanding candidates with a Ph.D. in any area of computer science and engineering will be considered. Successful candidates are expected to pursue an active research program, perform both graduate and undergraduate teaching, and supervise graduate students. The School offers both undergraduate honours programs and graduate programs up to the Ph.D. level. The programs have restricted enrollment in order to maintain a high quality research and teaching environment. Prospective faculty can look forward to a dynamic research environment in which collaboration with industry is encouraged. Carleton is located in Ottawa, the capital of Canada and a major center of advanced technology research and development.

Salary commensurate wtih qualifications and experience. Send curriculum vitae and names of three referees to Professor Frank Fiala, Acting Director, School of Computer Science, Carleton University, Ottawa, Ontario K1S 5B6, Canada.

Carleton University is committed to equality of employment for women, aboriginal peoples, visible minorities, and disabled persons. Interested persons from these groups are encouraged to apply. In accordance with Canadian immigration requirements, this advertisement is directed to Canadian citizens and permanent residents.

This position is subject to budgetary approval.

UNIVERSITY OF MINNESOTA
MINNEAPOLIS, MINNESOTA
Department of Electrical Engineering

The Department of Electrical Engineering invites applications and mominations for tenure track and tenured positions of all ranks. In addition for the 1991-92 academic year several visiting and temporary positions are available. The duties of tenure and tenure track faculty include the establishment of a sponsored research program and teaching at both the undergraduate and graduate level. Candidates with exceptional qualifications in all areas of electrical engineering will receive due consideration. Of particular interest, however, are individuals whose specialties lie in the areas of computer architecture and digital electronics, circuits and VLSI design, and microelectronics, with particular interests in optoelectronics, or micromachining.

The Department of Electrical Engineering has 46 faculty members providing undergraduate and graduate education and pursuing research in all areas of electrical engineering. The State-funded Microelectronics and Information Sciences Center and the Supercomputer Institute at the University of Minnesota provide opportunities for interdisciplinary research. The Department has a complete integrated circuit fabrication and design laboratory. Faculty have direct access to the full range of computing facilities including supercomputers.

The requirements for all positions include an earned Doctorate in an appropriate discipline at the time of the appointment and outstanding academic and research records. Rank and salary will be commensurate with the qualifications and experience of the selected candidates. Applications and nominations should be sent with a resume containing the names of three references to:

Professor Larry L. Kinney,
Chairman of the Faculty
Recruiting Committee,
Department of Electrical Engineering,
University of Minnesota,
200 Union St. SE, Minneapolis, MN
55455

The last date for receiving applications will be August 30, 1991 for positions available September 16, 1991. Review and interviews may begin as early as November 15, 1990, and early decisions can be made on positions.

The University of Minnesota is an equal opportunity educator and employer and specifically invites and encourages applications from women and minorities.

CASE INSTITUTE OF TECHNOLOGY
CASE WESTERN RESERVE
UNIVERSITY

We invite applications for tenure track faculty positions at all levels. Candidates from all research areas will be considered, but the thrust research areas in the Department are VLSI systems and design automation, applied artificial intelligence and logic programming, database design and systems, and software systems and design environments. Candidates should have a Ph.D. in computer science or computer engineering or closely allied fields; competitive salaries will be offered to attract the best candidates.

CWRU is a private university with a total enrollment of 8,400, of which 5,100 are graduate and professional students. The Engineering School of Case Institute of Technology is among the top ten engineering schools in terms of research funding per faculty member and undergraduate student quality. The University campus is the hub of the pleasant area known as University Circle, an incorporation with neighboring cultural centers and museums, about five miles from downtown Cleveland.

The Department of Computer Engineering and Science has 14 faculty positions, and a graduate student body of 110 students, 40 of which are in the Ph.D. program. Departmental facilities are based upon an ethernet local area network, connected to INTERNET, which supports a UNIX operating system and about 40 SUN and other workstations. In addition, faculty and students participating in the Center for Automation and Intelligent Systems Research have access to the Center's computing facilities.

The Department recently acquired the Nord Professorship, supported by a donation of over one and a half million dollars, for which we invite distinguished senior faculty applicants. This position will provide additional funds for travel, graduate student support and equipment.

Applicants should submit their curriculum vitae and names of at least three references to: Lee J. White, Chairman, **Department of Computer Engineering and Science,** Case Western Reserve University, Cleveland, Ohio 44106; INTERNET: leew@alpha.ces.cwru.edu; candidates with previous academic experience may wish to provide at most three reprints of their most important publications.

An equal employment and affirmative action employer.

HARVEY MUDD COLLEGE
Senior Position in Computer Science

Harvey Mudd College has re-opened its search for a senior professor of computer science to lead the department and help design a curriculum for a major in computer science. The successful candidate for this position will be able to provide leadership and direction both for the college's overall computer science program and for the new major. He or she must also desire the challenge and opportunity to develóp a program which builds on the existing strengths of the college in science and engineering. Qualifications for the position include a doctorate in computer science or in a related field with computer science experience, demonstrated commitment to excellence in teaching, a willingness and ability to participate in curriculum development, and significant professional experience.

Harvey Mudd College, one of the nation's most selective undergraduate colleges of engineering and science, is a member of the Claremont Colleges. The college is an equal opportunity/affirmative action employer. Please send resume and names of four references to: Nathaniel Davis
Dean of Faculty
Harvey Mudd College
Claremont, CA 91711

SOFTWARE ENGINEER

Responsible for the software engineering of process control systems upgrades and specials for paper applications for mftr. in Central Ohio. Analyzes system specifications & application processes; develops software scope & designs software subsystems; generates programs in any required language; selects & applies reusable program modules to meet project requirements; produces test plans & conducts tests of functional subsystem; performs full system functional verification and field testing; produces project documentation & software documents for technical users; participates in the preparation of end-user sysem documentation. Must have a B.S. in computer science and 3 years exp. in the job described or as systems engineer. As part of exp., must have 1) participated (6 months) in the design and development of a project in object-oriented programming in a real-time environment, 2) written 2000 assembly lines on 8080/8085, 3) field tested a software system with a minimum value of $1 million, 4) worked (1 year) on a system having redundancy features, 5) designed a serial link protocol with the following features: frame check sum, acknowledge/not acknowledge. Salary: $42,000, 40 hrs/wk. (Mon-Fri, 8AM-5PM). Must have proof of legal authority to work permanently in U.S. Send resume in *duplicate* (no calls) to S. Holton, JO# 1260209, Ohio Bureau of Employment Services, P.O. Box 1618, Columbus, Ohio 43216.

COMPUTER HARDWARE ENGINEER

Computer Hardware Engineer to evaluate computer systems Hardware and Software; quality control; analyze computer systems problems; hardware development, plan layout of new system installation or modification of existing system; analyze data to determine, recommended and plan layout for type of computer and peripheral equipment or modification of existing equipment and system, may specify power supply requirements. Salary for a 40 hr work week, 9am-5pm Mon-Fri is $600.00 weekly. Applicants with a B.S. degree in Electrical Engineering and two years exp. send resumes only to Job Service of Florida, 701 S.W. 27th Ave., Room 15, Miami, Florida 33135 Ref: Job Order # FL 0315748.

UNIVERSITY OF CALIFORNIA,
SANTA CRUZ

Computer Engineering at UC Santa Cruz expects to recruit for two faculty positions for the 1990-91 academic year: one Associate or Full Professor position, and one Assistant position. The areas of specialization have not been defined at this time.

For more information please see our ad in next months Computer Magazine or inquire at the following address: Chair, Computer Engineering Faculty Search Committee, Baskin Center for Computer Engineering & Information Sciences, Applied Sciences Building, University of California, Santa Cruz, CA 95064. UCSC is an EEO/AA/IRCA employer. (Via email to recruit@saturn.ucsc.edu or recruit@ucsccrls.bitnet)

STANFORD UNIVERSITY
Engineering Research Associate
Entry Level

Position available in wide spectrum specification languages and support environments for systems design and specification, involving design of new specification languages for concurrent and time-critical systems, and development of application tools. Will take responsibility for ongoing projects to develop software tools supporting applications of specification languages to production of multi-processor Ada systems. Projects require the employee to develop new software tools for which there are no previous paradigms, to write user manuals, to publish papers in archival scholarly journals, and to carry out integration of the new tools into commercially available fielded Ada programming environments.

Required: Ph.D. in Computer Science or Electrical Engineering; knowledge of Ada programming constructing Ada compilers and realtime multi-tasking Ada systems; knowledge of recent developments in Specification Languages, Program Verification, and Automated Deduction. Salary $55,000 annum. Place of employment and interviews: Stanford. Send this ad and a resume to Job #NOF 927, P.O. Box 9560, Sacramento, CA 95823-0560 no later than November 15, 1990.

RENSSELAER POLYTECHNIC INSTITUTE
Chairman of
Computer Science Department

Applications and nominations are invited for the position of Professor and Chairman of the Computer Science department. Candidates should be internationally recognized for their research, with an outstanding record of research support and strong commitment to excellence in teaching and service.

The department has twenty-two faculty members who conduct research in parallel and distributed computation, database systems, computer vision and image processing, learning algorithms, visual programming languages, VLSI systems, symbolic and numeric computation and theory of computation. The department has active undergraduate and graduate programs that offer B.S., M.S. and Ph.D. degrees in computer science. There are 300 undergraduate majors and 118 graduate students, including 46 Ph.D. students. The research program is supported by the government and major corporations at the level of 2 million dollars per year. In 1989 the department was granted an Institutional Infrastructure award from NSF. Computing facilities are excellent and support the full range of paradigms for parallel computing, symbolic computation and image processing, e.g., MASPAR, Intel hypercube, Sequent, Ardent, AIS-5000, Symbolics, and networks of Sparc, Dec and Macintosh work stations.

Various research projects are conducted in cooperation with other RPI departments, like Electrical, Computer and System Engineering or Applied Mathematics, and with Rensselaer's interdisciplinary centers for interactive computer graphics, integrated electronics, manufacturing productivity and applied mathematics and advanced computation. Our location in the Northeast is close by to many major computer companies, with many opportunities for collaboration. In particular, the corporate research and development center of General Electric is located within ten miles of Rensselaer. There is also an active incubator program which helps Rensselaer faculty and students launch start-up companies.

Rensselaer, the oldest technological university in the US, has been traditionally recognized for its excellence in teaching. This strong commitment to teaching has been most recently evidenced in the newly formed center for innovation in undergraduate education. Finally, the capital district region of New York provides an excellent living environment with many nearby recreational opportunities and close proximity to the major Eastern metropolitan areas.

Applicants should send a resume and the names of at least three references to: Professor Boleslaw Szymanski, Chairman of Search Committee, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, N.Y. 12180-3590 (telephone: 518-276-2714, email: szymansk@turing.cs.rpi.edu, fax: 518-276-4033). The position will be filled no later than July 1, 1991.

RPI is an Affirmative Action, Equal Opportunity Employer.

ASTEM, KYOTO, JAPAN
Advanced Software and Mechatronics Research Institute

The newly established ASTEM, Kyoto, Japan, invites applications for research associate and senior research positions.

Speciality areas include, but are not limited to, Software Engineering, Knowledge Base, Computer Graphics, and CAD/CAM/CIM. For a research associate position, a candidate should be highly qualified and hold an MS in CS, EE, or related fields; for a senior research position a PhD is normally required.

ASTEM provides an excellent environment for research in Software Engineering and AI with unique Japanese software and the state of the art workstations.

ASTEM is located in Kyoto, the ancient beautiful city of Japan.

Send resume to: Mr. A. Kamei,
ASTEM, Kyoto Research Park, 17 Chudoji, Minami-machi, Shimogyo, Kyoto 600 JAPAN
or call (213) 544-1103.

EMPLOYMENT OPPORTUNITY

Company needs an applicant with Master's Degree in Computer Engineering and having taken courses in network, math statistics, system programming and communication systems. Will do research, develop and implement communication network software systems, and to interface computer hardware with communication equipment. Salary $33,500/year, 40 hour/week. Send your resume to Mrs. Jimmie Gaston, Division of Employment Security, 505 Washington, St. Louis, Missouri 63101. Re: Job #410397.

UNIVERSITY OF CALGARY
Assistant Professor
Department of Computer Science

The University of Calgary Department of Computer Science invites applications for an Assistant Professor (tenure-track) effective July 1, 1991. Duties include teaching at the undergraduate and graduate levels (PhD. and MSc) and maintaining an active research program in computer science. A researcher in software engineering, systems or computer graphics is particularly required, however all qualified candidates in all areas of core computer science are encouraged to apply. Research facilities include a large network of Sun and Apollo workstations and fileservers connected using FDDI running Unix. Requirements include a PhD in computer science or a related area.

In accordance with Canadian immigration requirements, priority will be given to Canadian citizens and permanent residents of Canada. The University of Calgary has an Employment Equity Program and encourages applications from all qualified candidates, including women, aboriginal people, visible minorities, and people with disabilities.

Interested candidates should direct applications, including a curriculum vitae, copies of their three most important publications, and the names of three references, prior to January 31, 1991, to:

Dr. Jon Rokne, Head
Department of Computer Science
The University of Calgary
2500 University Drive N.W.
Calgary, Alberta, Canada
T2N 1N4
Telephone: (403) 220-5454

UNIVERSITY OF EVANSVILLE
Faculty Positions in Computer Science

The Department of Electrical Engineering and Computer Science at the University of Evansville invites applications for a tenure track faculty position in computer science for the fall semester of 1991. Candidates should hold a Ph.D. (or a Master's degree with at least four years' experience). All specialty areas of computer science will be considered. Salary and rank will be commensurate with academic background and experience. The University of Evansville is an undergraduate teaching institution with 2100 full-time undergraduates. Special consideration will be given to candidates with undergraduate computer science credentials and advanced degree work in either computer science or engineering. Applicants should be interested in helping develop ABET/CSAB accreditable programs. We encourage applications from qualified women and under-represented minorities. The University of Evansville is an equal opportunity/affirmative action employer. Please send resume and three references to: Dr. John R. Tooley, Dean of the College of Engineering and Computer Science, 1800 Lincoln Avenue, Evansville, Indiana 47722. Consideration of applications will begin on September 1, 1990, and continue until the position is filled.

ROYAL INSTITUTE OF TECHNOLOGY (KTH)
School of Computer Science and Engineering

Applications are invited for the position of Professor in Software Engineering, seen as the discipline studying methods and tools for the construction of large computer programs. Topics of interest include requirements specification, prototyping, programming, verification, testing, maintenance, and computer based tools.

Duties of the new professor will also include teaching at the undergraduate and graduate levels.

Candidates are invited to submit their application, addressed to the Swedish government, before October 31, 1990, to the Registrar, The Royal Institute of Technology (KTH), S-10044 Stockholm, Sweden. The application should include the reference number 797/90, a curriculum vitae and a description of research activities and plans.

For futher information please contact Prof. Lars-Erik Thorelli, dean of the School of Computer Science and Engineering, KTH, S-10044 Stockholm, Sweden, phone +46-8-7907880, fax +46-8-247784, email le@tds.kth.se.

UNIVERSITY OF NOTRE DAME
Chairperson
Department of Computer Science and Engineering

The University of Notre Dame invites applications for the Chair of the recently established Department of Computer Science and Engineering in the College of Engineering. This position is a unique opportunity to provide significant input in establishing direction for the initiation and growth of programs in this department. Persons with established records in education and scholarship who would welcome the challenge of developing a new department and its programs are encouraged to apply. The faculty of the department is expected to determine the curricula leading to graduate and undergraduate degrees in computer science and in computer engineering.

The Department of Computer Science and Engineering has twelve tenure-track faculty positions. The Department of Electrical Engineering and its present graduate and undergraduate options of emphasis in computer engineering are expected to provide the foundation for the programs in the new department. It is anticipated that the Department of Computer Science and Engineering will have close cooperative ties with the Department of Electrical Engineering, Department of Mathematics, and other academic units.

Qualified applicants are encouraged to submit a resume and letter of interest to:

Dr. Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

For additional information, please contact the Dean's office at (219) 239-5534.

The University of Notre Dame invites applications from all qualified persons without regard to sex, ethnic origin, religious preference or physical impairment.

ACADEMIA SINICA
Taiwan, Republic of China
Institute of Information Science

Applications are invited for research position in Institute of Information Science, Academia Sinica. Ph.D. in Computer Science or closely related fields required. Demonstratable research ability necessary. Applicants for senior positions must have proven research record. All fields in Computer Science are welcome.

The Institute offers a good research environment. No duty of teaching. Facilities include a 32-node NCUBE 2 parallel supercomputer, many SUN, SGI, and E&S workstations. An easily accessible ETA-10Q supercomputer is in the Academia Sinica.

Interested people please send application to Dr. Y.S. Kuo, Acting Director, Institute of Information Science, Academia Sinica, Taipei, Taiwan, 11529, Republic of China. Fax: (001-886-2) 782-4814.

THE UNIVERSITY OF CHICAGO
Department of Computer Science

Junior and Senior positions available. Our preference is for candidates with expertise in one of the areas of experimental Computer Science, such as Programming Languages or Distributed Systems, but we will consider exceptionally strong applicants from all areas. Send curriculum vitae and three letters of reference to: Prof. Janos Simon, Chairman, Department of Computer Science, The University of Chicago, 1100 E. 58th Street, Chicago, IL 60637. Inquiries can be directed to chair@cs.uchicago.edu.

The University of Chicago is an equal opportunity/affirmative action employer.

FLORIDA STATE UNIVERSITY
Department of Computer Science

The Department of Computer Science is seeking qualified candidates for expected tenure track positions. Areas of special interest include, but are not limited to, software engineering, operating system, networking, programming languages, computer graphics, database, and computer architecture.

Our department is a young and rapidly growing one. It offers B.S., M.S., and Ph.D. degrees, and it places a very strong emphasis on excellence in research and teaching.

Research facilities include a CRAY Y-MP/432, a CM2 Connection Machine, an ETA-10, internet connections through SURANET and ESnet, a group SUN 3 and SUN 4 workstations and servers, a VAX 11/780, and a Symbolics 3640.

Applicants should have a Ph.D. in computer science or a closely related area, and a strong commitment to teaching and research.

Send a resume and names of at least three professional references to:

Dr. Abe Kandel, Chairman
Department of Computer Science B-173
Florida State University
Tallahassee, FL 32306-4019

Florida State University is an Equal Opportunity Affirmative Action Employer.

GRADUATE RESEARCH ASSISTANTSHIPS

The Department of Ocean Engineering at Florida Atlantic University has a number of graduate research assistantships available to qualified students. Positions in the areas of marine structures, marine materials and corrosion, acoustics and vibrations, hydromechanics and marine vehicles will start in the Spring 1991 term. FAU is an Equal Opportunity Affirmative Action Employer, and offers the M.S. and Ph.D. in Ocean Engineering. Interested students should write to Dr. Stanley E. Dunn, Department of Ocean Engineering, Florida Atlantic University, 500 N.W. 20th Street, Boca Raton, FL 33431.

CALIFORNIA STATE UNIVERSITY, FULLERTON
Tenure-Track and Full-Time Lecturer Faculty Positions
Computer Science Department

The California State University, Fullerton (CSUF) seeks faculty candidates for tenure-track and full-time lecturer positions in the Computer Science (CS) Department effective as of August 19, 1991. CSUF is an urban university located in the heart of Orange County. The aerospace and computer industries create a professionally challenging environment for computer scientists. The department has an enrollment of about 1000 students and a team of 14 full-time and 20 part-time faculty members. The department's strong instructional programs lead to bachelor's and master's degrees as well as providing courses for non-majors.

The CS Department is one of the four departments in the School of Engineering and Computer Science. The other departments are Civil Engineering, Electrical Engineering, and Mechanical Engineering Departments. The school has an enrollment of about 2500 students. The school's new building provides substantial laboratory resources for teaching and research in computer science and has a rich and diverse complement of computer hardware.

The faculty candidates should have a doctorate in computer science or a closely-related discipline. They must be prepared to teach a variety of courses and specialize in one of the following subjects: operating systems and architecture, networks, database, or computer graphics. For promotion and/or tenure consideration, an excellent teaching record and a strong background in research and refereed publishing are required.

Applicants should indicate if they are interested in tenure-track or full-time lecturer position. Qualified applicants will be selected from applications received before February 1, 1991, or until the positions are filled. The university serves an ethnically diverse community. Minorities and women are encouraged to apply. For more information, and to request an application form, please contact Dr. Charles Mosmann, Chair, Computer Science Department, Room CS-522, California State University, Fullerton, Fullerton, California 92634; telephone number (714) 773-3700.

Affirmative Action Equal Opportunity Title IX Employer.

DISTRICT PROFESSIONAL SERVICES MANAGER

District Professional Services Manager, Atlanta. Manage, direct, develops Professional Services Department for SE District. Formulate Department revenue forecasting, budgeting, district strategy, planning/direction: Liaise with corporate marketing and product management. Supervise 8 professional employees and indirectly supervise 8 Systems Engineers; responsible for all personnel decisions. Direct pre and post sales system consultations. Design information systems specific to customer needs; plan system implementation. Responsible for long range project with major customer. Liaise with managers in North America, UK and Mexico; direct efforts with major software/hardware companies; coordinate development groups in North America and UK. Assess product/services and recommend on efficiency/profit. Assess Operations strategies/objectivies. Ensure adequate staffing; recruit staff and assign to projects/customer. BS in Science, Computer Science or Information Systems and 3 years experience in job offered or 3 years experience as Manager Information Systems required. Experience must include primary responsibility for corporate department/division with information systems manufacturer and formulation responsibility for business plans/budgetary goals. Requires experience supervising professional information systems staff; must have experience in software operation and implementation of ICL software and systems. 40 hrs./wk.; 8am-5pm; $60,000/yr. Apply in person or by resume to Georgia Department of Labor, 2972 Ask-Kay Dr., Smyrna, GA 30082 or to nearest Georgia Job Service Center. Reference Control #GA5439540.

THE UNIVERSITY OF TEXAS AT ARLINGTON

The Department of Computer Science Engineering at The University of Texas at Arlington invites applications for tenure-track or visiting faculty positions in all areas of computer science or computer engineering. Applicants with expertise relating to reliable real-time distributed systems, telecommunications software, object-oriented systems, scientific visualization, knowledge-based systems, or parallel processing will be given preference. Rank is open. An earned doctorate or equivalent and a commitment to teaching and scholarly research are required. Openings are expected for January and September 1991. Applications received prior to October 15, 1990 and March 1, 1991 will receive full consideration for January and September openings, respectively. Interested persons should send a resume and a list of references to Bill D. Carroll, Professor and Chairperson, Computer Science Engineering Department, P.O. Box 19015, The University of Texas at Arlington, Arlington, TX 76019. Phone: 817-273-3785. FAX 817-273-2548, Internet: carroll@evax.arl.utexas.edu.

The University of Texas at Arlington is an Equal Opportunity Affirmative Action Employer.

National University of Singapore

Department of Electrical Engineering

The Department of Electrical Engineering invites applications for **teaching and research positions** from candidates with a PhD degree in one of the following areas:

Computer Communications
Computer Systems
Image Processing/Computer Vision

Gross annual emoluments range as follows:

Research Fellow	S$44,860 - 64,200
Lecturer	S$53,160 - 64,200
Senior Lecturer	S$58,680 - 100,310
Associate Professor	S$88,650 - 122,870

(US$1.00 = S$1.81 approximately)

The commencing salary will depend on the candidate's qualifications, experience and the level of appointment offered.

Leave and medical benefits will be provided. Depending on the type of contract offered, other benefits may include: provident fund benefits or an end-of-contract gratuity, a settling-in allowance of S$1,000 or S$2,000, subsidised housing at nominal rentals ranging from S$100 to S$216 p.m., education allowance for up to three children subject to a maximum of S$10,000 per annum per child, passage assistance and baggage allowance for the transportation of personal effects to Singapore. Staff members may undertake consultation work, subject to the approval of the University, and retain consultation fees up to a maximum of 60% of their gross annual emoluments in a calendar year.

The Electrical Engineering Department has currently an academic staff of 51 with 18 laboratories, all of which have excellent facilities for teaching and research. Its Semiconductor Laboratory has a Riber 32P Molecular Beam Epitaxy System and 2 liquid phase epitaxy systems for research into III-V compound devices, while its Microelectronics Laboratory has facilities for fabricating ICs in 4-micron poly-gate single metal CMOS technology. A wide range of computing resources are available, including numerous PCs, SUN Sparcstations, Microvaxes, and HP 9000 Series 300s. The University Computer Centre operates an IBM3081 KX2, and is acquiring a high-speed campus-wide network directly linking the staff's PCs (now provided to every staff member) to the various computing resources, including 2 supercomputers based in the nearby Science Park. A number of large-scale research projects are in progress, including an optical LAN joint effort with Singapore Telecoms and project to develop VLSI design tools jointly with Chartered Semiconductors.

Application forms and further information on terms and conditions of service may be obtained from:

The Director
Personnel Department
National University of Singapore
10 Kent Ridge Crescent
Singapore 0511

The Director
North America Office
National University of Singapore
55 East 59th Street
New York, NY 10022 U.S.A.
Telephone: (212) 751-0331

Enquiries on current research activities in specific areas may be sent through BITNET to: **PERLT @ NUS3090**

BOOK REVIEWS

Editor: Guy Johnson, Department of Information Technology, Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY 14623.

Computer Graphics

Francis S. Hill, Jr. (Macmillan, New York, 1990, 754 pp., $51)

By taking an engineering approach to this introduction to computer graphics, Hill has produced a solid text for a one- or two-semester course on computer graphics concepts and techniques at the junior, senior, or graduate level. For a core course in graphics, it is definitively a more up-to-date alternative to such classics as Newman & Sproull's *Principles of Interactive Computer Graphics* (McGraw-Hill, 1979) and Foley & van Dam's *Fundamentals of Interactive Computer Graphics* (Addison-Wesley, 1982). The book is especially noteworthy for its outstanding introduction to ray tracing.

The book is targeted at fields ranging from computer science to the arts, but it is most appropriate for electrical engineering, computer science, physics, business, and art majors, as well as math undergraduates. Its lack of formal theory and advanced topics to back up the many code segments and examples might leave computer science or mathematics graduate students wanting more. However, this lack makes the book accessible to general scientific audiences and provides an excellent implementation reference for software engineers. Readers should have experience with elementary data structures and Pascal, C, or Fortran, as well as college-level algebra, geometry, trigonometry, calculus, and matrix theory.

The book's scope is quite impressive, but not comprehensive. It covers the field from graphical hardware devices to advanced rendering techniques. Most of the book consists of basic material presented intuitively, and its organization reminds me of the quick-and-dirty how-to books I endured during my engineering years. The text is illustrated by plenty of examples and code segments in pseudo-Pascal.

Each chapter begins with a statement of goals, contains many drills and programming exercises, and ends with a summary and more exercises. (The entire text has more than 440 drill exercises and about 180 programming problems.) Chapters 3-6 thoroughly examine the fundamentals of the field, including windows, viewpoints, clipping, polygon drawing, curves, and fractals. After this highly practical material, Chapter 7 returns to the basics of geometry to prepare the reader for the later material on ray tracing. Hill then exhaustively describes patches, surfaces of revolution, quadrics, and superquadrics to give the reader some concrete examples of geometry, something most computer graphics texts don't have.

Hill's introduction to ray tracing is the most complete introduction in textbooks today. Nowhere else except in conference tutorials or monographs has the topic been covered as well.

Hill then introduces 2D and 3D affine transformations, along with homogeneous coordinates and such uncommon transformations as fish-eye and unit-circle inversion. His coverage of projections is the best I have seen and includes a complete taxonomy of projections and an excellent model of 3D viewing with the synthetic camera. The sections on raster techniques and surface design are very accessible but not very deep; for instance, they omit the theory behind Bezier surfaces. Hill then takes the reader through classic shading methods to hidden surface elimination, such as the z-buffer and the Warnock algorithm techniques. Unfortunately, he does not cover all HSR/HLR methods and does not mention any object space algorithms, which are the most conceptually interesting.

It's Chapter 18 that should bring this book the attention it deserves. Hill's introduction to ray tracing is the most complete introduction in textbooks today and offers many examples, code segments, and exercises. I highly recommend the book to everyone frustrated by previous literature that left them in the dark. Nowhere else except in conference tutorials or monographs has the topic been covered as well. The book's value as a starting point for research is limited, though, since it has very few references throughout the text and lacks selected bibliographies at the end of each chapter.

The book closes with material on advanced clipping algorithms and description of several standards. However, Hill's focus on fundamentals leaves several important areas unexplored, such as modeling, advanced rendering, special effects, animation, natural phenomena simulation, and artistic points of view. Most contemporary textbooks — such as Rogers' *Procedural Elements for Computer Graphics* (McGraw-Hill, 1985), Magnenat-Thalmann and Thalmann's *Image Synthesis: Theory & Practice* (Springer-Verlag, 1987), and Burger & Gillies' *Interactive Computer Graphics* (Addison-Wesley, 1989) — discuss such topics extensively.

A.M. Baldassarre
University of Houston

Elements of Computer Music

F. Richard Moore (Prentice Hall, Englewood Cliffs, N.J., 1990, 560 pp., $33.40)

My ideal "book" on computer music would be an interactive, multimedia treatise in which musical examples could be heard as well as seen, programs could be run as well as listed, and the text could be heard as well as read. It shouldn't be long before such contraptions become readily available. In the meantime, students of computers and music for the most part must settle for old-fashioned books. Moore's *Elements of Computer Music* lessens one's impatience for the newfangled kind.

Moore has directed the Center for Music Experiment at the University of California at San Diego since 1974, and he has a long and varied history in the field. His Cmusic — an "acoustic compiler" — is the first general music program written in a language that is almost universally available — in this case, C.

Moore's book is divided into chapters on "digital audio," "instruments," "rooms," and "composing." His presentation of the basic concepts and methods in each of these areas is cogent and is profusely illustrated with Cmusic program examples. As needed, Moore covers the main points of the physics of sound, psychoacoustics, and digital signal processing. Each chapter ends with a number of appropriate references, making the book a useful guide to computer music literature.

Although they are sometimes peripheral to the main text, interesting ideas and facts abound in this book. Moore draws an interesting analogy between the computer and the telescope (both technological artifacts offering dramatically closer looks at important phenomena), discusses Respighi's use of gramophonic nightingales in his 1925 *Pines of Rome*, observes that the ultimate instrument every performer plays is the room in which the music is being heard, and examines the relationship of temporal elements of music to the human body's characteristic periodicities. I was intrigued by the possibilities of composing in virtual acoustic space as well as in the traditional dimensions of time, pitch, and timbre.

This work is ambitious theoretically. Not many books on computers and music start their chapters with quotes from Schelling, Bacon, Kuhn, Plato, and the like. Although Moore is at his best when he stays close to his disciplinary home, his broader speculations underscore the humanistic and creative implications of music's encounter with modern machines.

Moore manages to maintain technical rigor without sacrificing readability, but those readers whose algebra and trigonometry aren't fresh could find themselves swamped (or forced to forego the extended mathematical excursions). There are some reassuring parenthetical comments ("Do not worry if all the implications of these formulas are not immediately obvious to you — that is what this section is about!") and a useful appendix of relevant mathematical ideas. If you have a basic knowledge of music, acoustics, programming, and mathematics, you should have few problems.

The book hardly purports to be an encyclopedia and does not avoid unevenness of coverage. It includes a refreshingly clear exposition of Markov chains and other stochastic processes, and a glancing but suggestive interconnection of fractals, power spectra, and Brownian motion. On the other hand, the MIDI standard is mentioned only once, and the book ends abruptly with two paragraphs on algorithmic composition, surely a topic deserving at least as much attention as random sieves (which get about 20 pages). Moore's lucid treatment of tuning and the fundamental problems of organizing pitch is relegated to an appendix (I have the feeling it was originally intended as a chapter), and the excellent chapter summaries stop short of the last chapter. Lastly, Moore provides a thorough index, but a glossary would have come in handy — especially for words like "agogic" and "aleatoric."

Despite these arguable flaws, *Elements of Computer Music* is a superb introduction for those "interested in learning how to use computers to extend the boundaries of music," as Moore puts it. It is a thought-provoking and informative treatment of a fascinating field that will increasingly affect us all — even those not eagerly awaiting interactive audiovisual devices that let us do fun things like creating and manipulating sonic events of arbitrary complexity in real time.

Marc Lauritsen
Harvard Law School

Data Exchange PC/MS DOS

Steven S. Ross (McGraw-Hill, New York, 1989, 411 pp., $27.95)

This book is intended to explain methods for exchanging files between various word processors, spreadsheet programs, or database applications to business professionals who are not computer literate. However, the book is too verbose for the material it covers. It could be half the size and cover the same material more effectively.

Ross does a good job of describing the file characteristics and idiosyncrasies of most of the significant word processors, spreadsheets, and databases, including Wordstar, Xywrite, Volkswriter, Emacs, Microsoft Word, Word Perfect, Lotus, Symphony, Visicalc, Quattro, dBase, and Supercalc. However, he should have summarized the file characteristics in two or three chapters — one each for word processors, spreadsheets, and databases — instead of 14.

Ross explains the file attributes that need to be changed to convert a file between different word processors. The user can either do the conversion manually (change each required attribute using an editor — tedious, but often the only feasible method), write a utility, or buy one. Ross includes some elementary examples of how to write a file-conversion utility and provides a good list of existing programs. He does not make any recommendations, though, nor does he indicate if any of the utilities require the user to do further editing.

The book includes a decent list of media conversion programs, as well as information on cables to connect different PC architectures, for example, IBMs and Macintoshes. However, Ross does not explain how to set baud rates, bit rates, stop bits, or other parameters, nor does he discuss how to convert DOS and Apple files that have different file attributes. He also does not give any information on converting between PCs and Unix, VAX/VMS, or Sun systems, or on such utilities and drivers as Kermit and Crosstalk.

Krishna Vallabhaneni
Milpitas, California

Software Quality Concepts and Plans

Robert H. Dunn (Prentice Hall, Englewood Cliffs, N.J., 1990, 296 pp., $51.50)

As the title implies, this book explores software quality. The author has collected the material into five loosely assembled sections. I say "loosely" because the sections are defined more by individual focuses than by strict divisions.

The first part — Chapters 1 and 2 — gives a "plain vanilla" overview of software quality and introduces the balance of the book. The second part comprises Chapters 3-5 and identifies three elements of quality: people, technology, and management. In the third part, Chapters 6 and 7 define software quality programs by looking at software development schemes and the "hooks" for ensuring quality. Chapter 8 is devoted to maintenance and its ramifications.

The fourth part discusses quality management and is the best of the book. In Chapter 9, Dunn restates that "software quality assurance is the management of a software quality program." The chapter deals with the software quality organization, employee roles, and defect correction and tracking. Chapter 10 then zeros in on "measurement and methods used as part of a quality improvement program" — that is, the analysis of defect data to identify areas in which process improvement is likely.

A good share of Dunn's audience might find the last section worth the price of the whole volume. Chapters 11-14 present an overview of software quality plans and actual samples of three types of plans: a plan for a small project, a plan for a large project (based on ANSI/IEEE Std. 730-1984), and a plan in accordance with DoD-Std-2186. These should give almost everyone an example of a software quality plan to fit their needs.

Dunn recognizes that some of his audience will not need to read all the chapters, and he notes (correctly) that some chapters are too basic for some readers or too technical for others. Therefore, the excellent chapter summaries are a major strength of the book and will save many readers a good deal of boredom or confusion, depending on their experience level.

I review a text by reading the whole book and making comments in the margins for use as I write the review. This book's effect on me is apparent in its very clean margins. There is little of note, positive or negative. However, I did find the first three sections vague, bland, and rambling. A true software quality neophyte might find the material useful as a general introduction to Dunn's view of the topic. The experienced analyst can read the chapter summaries and find pointers to the occasional, hidden bits of wisdom.

The presentation in the first part is ill focused and refers often to other sections of the text. I also found that my frequent need to refer to a dictionary interrupted my train of thought. I like to think I have a rather robust vocabulary, but words like "cachet," "recondite," "concomitant," "ontogeny," "anthropocentric," "locution," "iconoclasm," "tocsin," "verisimilitude," "sobriquet," and "palimpsest" seem better suited to a spelling bee than to a software quality text.

As a reward for fighting through the early chapters, Dunn offers the reader excellent reference lists at the end of each chapter, providing a wonderfully broad spectrum of current software quality literature.

If you are looking for a good book on software quality, I recommend Dunn and Ullman's *Quality Assurance for Computer Software* (McGraw-Hill, 1982). But if you need to see samples of software quality plans, or if you want insights into improving software processes, *Software Quality Concepts and Plans* is the book to buy.

John W. Horch
The Horch Company

Computers and Engineering Management

Thomas F. Wheeler (McGraw-Hill, New York, 1989, 369 pp., $46.50)

Engineering managers need a variety of information to help them make decisions about computer systems implementation. This book attempts to bring together at least some of that information. Unfortunately, it falls far short, resulting in a rather disappointing and commonplace collection of computer history, insights, and predictions.

The book is targeted at the mechanical or electronic engineering manager with a limited knowledge of computers and computer applications, but it fails to meet this audience's major needs. Also, its format makes it difficult to get detailed information or insights on specific topics. Wheeler's discussions often ramble, making it seem that he's forcing his personal experiences into the text at the expense of flow.

Although a fair number of figures and illustrations are scattered through the text, they do little to support or enhance it. The typical manager would find a greater number of tables and checklists more useful. Also, the single bibliography at the end contains few references to outside articles, papers, or books, making it difficult to find additional reading on specific topics.

Wheeler sometimes sacrifices accuracy for simplicity and generality. I noted several technical inaccuracies and omissions. For example, Wheeler defines "EPROMs" as an acronym for "electronic PROMs." Also, his discussion of human-interface input devices omits light pens and trackballs while devoting an entire paragraph to the problems of the IBM PC Jr. keyboard.

In several places, Wheeler also presents statistics without reference to supporting studies or papers. Several statements presented as fact are actually either opinions or unsubstantiated assertions.

This book will serve as neither a reference book nor a source book. It does not serve the needs of either computer-illiterate or -literate managers. I can only recommend it to readers interested in Wheeler's insights on computer history or his personal prophecies on the future of computing.

James B. McClanahan
Public Service of Oklahoma

Current contents CS MAGAZINES

September 1990 *IEEE Computer Graphics and Applications*

Techniques for Cubic Algebraic Surfaces, Thomas W. Sederberg

Recognizing Shape Features in Solid Models, Hiroshi Sakurai and David C. Gossard

Surface Mapping Brain Function on 3D Models, Bradley A. Payne and Arthur W. Toga

Vector versus Raster: A Functional Comparison of Drawing Technologies, Charles M. Eastman

Rendering Revisited

Introduction: Something Old, New, Borrowed, and Blue, Jack Bresenham

Accurate Rendering by Subpixel Addressing, Olin Lathrop, David Kirk, and Doug Voorhies

Line-Drawing Algorithms for Parallel Machines, Alex T. Pang

Parallelization of Bresenham's Line and Circle Algorithms, William E. Wright

September 1990 *IEEE Software*

Applications of Formal Methods: Developing Virtuoso Software, Susan L. Gerhart

Seven Myths of Formal Methods, Anthony Hall

Specifying a Real-Time Kernel, J. Michael Spivey

A Formal Specification of an Oscilloscope, Norman Delisle and David Garlan

Integrating Formal Methods into the Development Process, Richard A. Kemmerer

Formal Verification of a Pipelined Microprocessor, Mandayam Srivas and Mark Bickford

The Case for Formal Methods in Standards, David Blyth, Cornelia Boldyreff, Clive Ruggles, and Mik Tetteh-Lartey

Data-Parallel Programming on Multicomputers, Michael J. Quinn and Philip J. Hatcher

Generations of User-Interface Management Systems, Deborah Hix

For subscription information, circle number 200 on the reader service card.

Magazine order form

Please print.

Single issue prices:
Member, $10
Nonmember, $20

Name _____

Address _____

City _____

State/ZIP/Country _____

IEEE-CS member no. (required for discount) _____

Mail to: IEEE Computer Society Order Dept., 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA, 90720-1264.

Quantity	Price	Total
Sept. '90 *CG&A*		
Sept. '90 *Software*		
Total (payment enclosed)		

VISUALIZE THE POSSIBILITIES

Three New Journals from Academic Press

Sample copies and privileged personal rates are available upon request.
For more information, please write or call:

ACADEMIC PRESS, INC.
Journal Promotion Department
1250 Sixth Avenue
San Diego, CA
92101, U.S.A.
(619) 699-6742

All prices are in U.S. dollars and are subject to change without notice.

S1026A

Computer Vision, Graphics, and Image Processing
is expanding into two publications in 1991:

Editors-in-Chief
Norman Badler
University of Pennsylvania, Philadelphia
Rama Chellappa
University of Southern California, Los Angeles

CVGIP: Graphical Models and Image Processing

Area Editors: **Brian A. Barsky, Gabor Herman, R.L. Kashyap, Nelson Max, Arun N. Netravali, Franco P. Preparata, Aristides A. Requicha, Demitri Terzopoulos, John W. Woods**

CVGIP: Graphical Models and Image Processing focuses on the synthesis methods and computational models underlying computer-generated or -processed imagery.

Volume 53 (1991), 6 issues ISSN 1049-9652 In the U.S.A. and Canada: $200.00 All other countries: $237.00

Editor-in-Chief
Linda G. Shapiro
University of Washington, Seattle

CVGIP: Image Understanding

Area Editors: **Ruzena K. Bajcsy, Larry S. Davis, Herbert Freeman, Robert M. Haralick, Thomas S. Huang, Ramesh Jain, Edward Riseman, Azriel Rosenfeld, Hanan Samet, Steven L. Tanimoto**

The central focus of this journal is the computer analysis of pictorial information. **CVGIP: Image Understanding** publishes papers covering all aspects of image analysis from the low-level, iconic processes of early vision to the high-level symbolic processes of recognition and interpretation.

Volumes 53–54 (1991), 6 issues ISSN 1049-9660 In the U.S.A. and Canada: $264.00 All other countries: $302.00

Editors-in-Chief
Yehoshua Y. Zeevi
Technion-Israel Institute of Technology, Haifa and CAIP Center, Rutgers University, Piscataway, New Jersey
T. Russell Hsing
Bell Communications Research, Morristown, New Jersey

New in 1990 **Journal of Visual Communication and Image Representation**

This new journal publishes papers on the state-of-the-art of visual communication and image representation with emphasis on novel technologies and theoretical work in this multidisciplinary area of pure and applied research.

Volume 2 (1991), 4 issues ISSN 1047-3203 In the U.S.A. and Canada: $128.00 All other countries: $154.00

USE THE READER SERVICE CARD TO OBTAIN INFORMATION ON:

- Membership application, student #203, others #202
- Perodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail II electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204
  (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

IEEE COMPUTER SOCIETY

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Helen M. Wood*
National Oceanic and Atmospheric Administration
F8 4, Rm. 1069, Code E/SP
Washington, DC 20233
(301) 763-1564

President-Elect: Duncan H. Lawrie*
Past President: Kenneth R. Anderson*

VP, Conferences and Tutorials: Laurel V. Kaleda (1st VP)*
VP, Standards: Paul L. Borrill (2nd VP)*
VP, Area Activities: Gerald L. Engel†
VP, Education: Ronald G. Hoelzeman†
VP, Membership and Information: Barry W. Johnson†
VP, Press Activities: James H. Aylor†
VP, Publications: Sallie V. Sheppard*
VP, Technical Activities: Mario R. Barbacci*

Secretary: David Pessel*
Treasurer: Joseph Boykin†
Division V Director: Edward A. Parrish, Jr.†
Division VIII Director: J. T. Cain†
Executive Director: T. Michael Elliott†
*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1990:
Vishwani Agrawal, Mario R. Barbacci,
Ming T. (Mike) Liu, Yale N. Patt, Donald E. Thomas,
Benjamin W. Wah, Ronald Waxman

Term Expiring 1991:
P. Bruce Berra, Michael Evangelist,
Ted Lewis, Raymond E. Miller, Earl E. Swartzlander, Jr.,
Joseph E. Urban, Thomas W. Williams

Term Expiring 1992:
Alicja I. Ellis, Tadao Ichikawa,
David Pessel, Sallie V. Sheppard, Bruce D. Shriver,
Harold Stone, Wing N. Toy

Next Board Meeting
November 16, 1990, 8:30 a.m.
New York Hilton, New York, NY

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Administration: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan

COMPUTER SOCIETY OFFICES

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

Publications Office
10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office
13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office
Doshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 408-3118
Fax: 81 (3) 408-3553

IEEE OFFICERS

President: Carleton A. Bayless
President-Elect: Eric E. Sumner
Executive Vice President: Martha Sloan
Secretary: Fumio Harashima
Treasurer: Wallace S. Read

VP, Educational Activities: Richard S. Nichols
VP, Professional Activities: Michael J. Whitelaw
VP, Publication Activities: Ralph W. Wyndrum, Jr.
VP, Regional Activities: Robert T. H. Alden
VP, Technical Activities: H. Troy Nagle, Jr.

Preliminary CALL for PAPERS

OOPSLA '91

Sixth Annual ACM Conference on

Object-Oriented Programming Systems, Languages, and Applications

6-11 October 1991, Phoenix, Arizona

Program Committee

Alan Snyder, HP Labs (chair)
Grady Booch, Rational
Luca Cardelli, DEC SRC
Pierre Cointe, Rank Xerox
William Cook, HP Labs
Peter Deutsch, ParcPlace Systems
Gail Kaiser, Columbia U.
Henry Levy, U. Washington
Henry Lieberman, MIT
Mark Linton, Silicon Graphics
Ole Lehrmann Madsen, Aarhus U.
Ron Morrison, U. St. Andrews
Eliot Moss, U. Massachusetts
Mayer Schwartz, Tektronix
Guy Steele, Thinking Machines
Jacob Stein, Servio Corporation
Dave Thomas, Object Technology
Mario Tokoro, Keio U./Sony CSL
Chris Tomlinson, MCC
Dionysios Tsichritzis, U. Geneva
Jeannette Wing, CMU
Rebecca Wirfs-Brock, Tektronix
Aki Yonezawa, U. Tokyo

Program Chair

Alan Snyder
OOPSLA 91 Program Chair
Hewlett-Packard Laboratories
(1501 Page Mill Road)
P.O. Box 10490
Palo Alto CA 94303-0969
+1-415-857-8764
oopsla91@hplabs.hp.com

Conference Chair

John Richards
OOPSLA 91 Conference Chair
IBM Research Center
P.O. Box 704
Yorktown Heights NY 10598
+1-914-784-7731
oopsla91@ibm.com

acm®

Object technology is influencing many areas of computer technology, including programming languages, software engineering, user interfaces, databases, distributed systems, and OSI network management. The availability of practical support for applying object technology is encouraging its use in real-world software development.

The annual OOPSLA conference is the premier forum bringing together researchers, developers, and practitioners to share ideas and experiences related to object technology. Areas of interest include: language design and implementation, tools and environments, components and frameworks, principles and theory, concurrent and distributed systems, methods and processes, databases and persistence.

Papers

The conference will include both invited and contributed papers. Authors are encouraged to submit high quality papers describing relevant research or experience. **Research papers** should describe work whose purpose is to advance the state of the art of object technology. **Experience papers** should describe work in which object technology has been applied for some purpose. The program committee will evaluate each paper on its relevance, clarity, correctness, originality, and significance. Special consideration will be given to promising experience papers. Papers must not be under consideration elsewhere (or published) in the same or similar form. Do not submit multiple papers describing essentially the same work.

Authors should send **six** copies of the full paper, in English, to the program chair to be received no later than **1 March 1991.** Papers must be limited to 18 pages, typed double spaced or typeset 10-point on 16-point spacing. A separate cover sheet must contain contact information (contact name, postal address, and phone number), a 100-word abstract, and indicate the paper category (research or experience). *Submissions failing to meet these conditions will be rejected.*

Authors will be notified of acceptance or rejection by **20 May 1991**. Accepted papers prepared on special forms are due **20 July 1991**. Authors of accepted papers are expected to sign an ACM copyright release form and present the paper at the conference. Proceedings will be distributed at the conference and via *SIGPLAN Notices* and will be available from ACM Press. Outstanding papers may be considered for a special issue of a journal. Awards will be given to the best paper in each category.

Guidelines for authors will be published in the Fall 1990 issue of *OOP Messenger*, or can be obtained from the program chair.

Experience Reports

Space will be made available in a separate track for short presentations of unrefereed reports describing practical experience applying object technology to real-world, product-quality software development. Prospective speakers should submit a 1-2 page description of the project scope and status and the specific points to be covered in the presentation to the program chair by **1 April 1991**. Selection will be based on relevance and potential interest. Summaries will be collected after the conference for publication in *OOP Messenger*.

Other Contributions

For information on tutorials, panels, workshops, demos, and vendor exhibits, see the Call for Participation, which will be available in October at the 1990 conference or from the conference chair.

Important Dates

1 March 1991
Papers due
1 April 1991
Experience reports due
20 May 1991
Acceptance notification
20 July 1991
Camera-ready papers due

Papers due 1 March 1991

To open the window of opportunity in solving DSP and Image problems, open our window first.

HYPERSIGNAL-WINDOWS™

Comprehensive Advanced Signal Processing Environment for Engineers and Scientists.

Hyperception

For more information, including catalog, contact:

Hyperception, Inc.
9550 Skillman LB 125
Dallas, TX 75243

Phone: 214-343-8525
FAX: 214-343-2457

International Representatives

GERMANY - Electronic Tools, phone: (02102) 841013, TLX 1631 + BTX 02102841013 1 +, fax: (02102) 841000 * UK, IRELAND - Loughborough Sound Images, LTD., phone: (0509) 231843, TLX 341409 LUFBRAG. fax: (0509) 262433 * FINLAND - ITT, phone: (90) 739 100, TLX 121450 MultiKomponent, fax: (90) 712414 * FRANCE - BORES Technical Sales, phone: CC44 (0483) 740136 * DENMARK - Assentoft Electronics, phone: (86) 16 29 26, fax: (86) 16 20 12 * ISRAEL - IES Ltd., phone: (03) 7526333, fax: (03) 7510927 * TAIWAN, ROC - EXARTECH International Corp., phone: 5372201 ~ 3, fax: (02) 5422689, TLX:26173 EXARTECH

Reader Service Number 9